IgH EtherCAT master Reference Manual

1.0

Generated by Doxygen 1.4.6

Wed Aug 2 18:41:43 2006

Contents

Chapter 1

The IgH EtherCAT master

1.1 General information

This HTML contains the complete code documentation.

The API documentations are in the `modules section`.

For information how to build and install, see the INSTALL file in the source root.

1.2 Contact


```
Florian Pose <fp@igh-essen.com>
Ingenieurgemeinschaft IgH
Heinz-Baecker-Str. 34
D-45356 Essen
http://igh-essen.com
```


1.3 License

```
code, have to sign an agreement to guarantee that products using
software based on IgH EtherCAT master stay compatible with the actual
EtherCAT specification (which are released themselves as an open
standard) as the (only) precondition to have the right to use EtherCAT
Technology, IP and trade marks.
```

Chapter 2

IgH EtherCAT master Module Index

2.1 IgH EtherCAT master Modules

Here is a list of all modules:

Chapter 3

IgH EtherCAT master Directory Hierarchy

3.1 IgH EtherCAT master Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 4

IgH EtherCAT master Hierarchical Index

4.1 IgH EtherCAT master Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

IgH EtherCAT master Data Structure Index

5.1 IgH EtherCAT master Data Structures

Here are the data structures with brief descriptions:

Chapter 6

IgH EtherCAT master File Index

6.1 IgH EtherCAT master File List

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

Chapter 7

IgH EtherCAT master Page Index

7.1 IgH EtherCAT master Related Pages

Here is a list of all related documentation pages:

Chapter 8

IgH EtherCAT master Module Documentation

8.1 EtherCAT realtime interface

8.1.1 Detailed Description

EtherCAT interface for realtime modules.

This interface is designed for realtime modules that want to use EtherCAT. There are functions to request a master, to map process data, to communicate with slaves via CoE and to configure and activate the bus.

Functions

- int **ecrt_slave_sdo_read** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t ∗target, size_t ∗size)

  Reads a CANopen SDO in normal mode.

- int **ecrt_slave_sdo_read_exp8** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t ∗target)

  Reads an 8-bit SDO in expedited mode.

- int **ecrt_slave_sdo_read_exp16** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t ∗target)

  Reads a 16-bit SDO in expedited mode.

- int **ecrt_slave_sdo_read_exp32** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t ∗target)

  Reads a 32-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp8** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t value)

  Writes an 8-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp16** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t value)

Writes a 16-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp32** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t value)

  Writes a 32-bit SDO in expedited mode.

- **ec_slave_t** * **ecrt_domain_register_field** (**ec_domain_t** *domain, const char *address, const char *vendor_name, const char *product_name, void **data_ptr, const char *field_name, unsigned int field_index, unsigned int field_count)

  Registers a data field in a domain.

- int **ecrt_domain_register_field_list** (**ec_domain_t** *domain, const **ec_field_init_t** *fields)

  Registeres a bunch of data fields.

- void **ecrt_domain_queue** (**ec_domain_t** *domain)

  Places all process data datagrams in the masters datagram queue.

- void **ecrt_domain_process** (**ec_domain_t** *domain)

  Processes received process data.

- int **ecrt_domain_state** (**ec_domain_t** *domain)

  Returns the state of a domain.

- **ec_domain_t** * **ecrt_master_create_domain** (**ec_master_t** *master)

  Creates a domain.

- int **ecrt_master_activate** (**ec_master_t** *master)

  Configures all slaves and leads them to the OP state.

- void **ecrt_master_deactivate** (**ec_master_t** *master)

  Resets all slaves to INIT state.

- int **ecrt_master_fetch_sdo_lists** (**ec_master_t** *master)

  Fetches the SDO dictionaries of all slaves.

- void **ecrt_master_sync_io** (**ec_master_t** *master)

  Sends queued datagrams and waits for their reception.

- void **ecrt_master_async_send** (**ec_master_t** *master)

  Asynchronous sending of datagrams.

- void **ecrt_master_async_receive** (**ec_master_t** *master)

  Asynchronous receiving of datagrams.

- void **ecrt_master_prepare_async_io** (**ec_master_t** *master)

  Prepares synchronous IO.

- void **ecrt_master_run** (**ec_master_t** *master)

  Does all cyclic master work.

- **ec_slave_t** ∗ **ecrt_master_get_slave** (const **ec_master_t** ∗master, const char ∗address)

  Translates an ASCII coded bus-address to a slave pointer.

- void **ecrt_master_callbacks** (**ec_master_t** ∗master, int(∗request_cb)(void ∗), void(∗release_cb)(void ∗), void ∗cb_data)

  Sets the locking callbacks.

- int **ecrt_master_start_eoe** (**ec_master_t** ∗master)

  Starts Ethernet-over-EtherCAT processing for all EoE-capable slaves.

- void **ecrt_master_debug** (**ec_master_t** ∗master, int level)

  Sets the debug level of the master.

- void **ecrt_master_print** (const **ec_master_t** ∗master, unsigned int verbosity)

  Outputs all master information.

- **ec_master_t** ∗ **ecrt_request_master** (unsigned int master_index)

  Reserves an EtherCAT master for realtime operation.

- void **ecrt_release_master** (**ec_master_t** ∗master)

  Releases a reserved EtherCAT master.

- int **ecrt_slave_write_alias** (**ec_slave_t** ∗slave, uint16_t alias)

  Writes the "configured station alias" to the slave's EEPROM.

- int **ecrt_slave_field_size** (**ec_slave_t** ∗slave, const char ∗field_name, unsigned int field_index, size_t size)

8.1.2 Function Documentation

8.1.2.1 int ecrt_slave_sdo_read (ec_slave_t ∗ *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint8_t ∗ *target*, size_t ∗ *size*)

Reads a CANopen SDO in normal mode.

Returns:
0 in case of success, else < 0

Todo
Make size non-pointer.

Parameters:
slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

target memory for value

size target memory size

Definition at line 191 of file canopen.c.

8.1.2.2 int ecrt_slave_sdo_read_exp8 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint8_t * *target*)

Reads an 8-bit SDO in expedited mode.

See **ec_slave_sdo_read_exp()**(p. 67)

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
sdo_index SDO index
sdo_subindex SDO subindex
target memory for read value

Definition at line 603 of file canopen.c.

8.1.2.3 int ecrt_slave_sdo_read_exp16 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint16_t * *target*)

Reads a 16-bit SDO in expedited mode.

See **ec_slave_sdo_read_exp()**(p. 67)

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
sdo_index SDO index
sdo_subindex SDO subindex
target memory for read value

Definition at line 624 of file canopen.c.

8.1.2.4 int ecrt_slave_sdo_read_exp32 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint32_t * *target*)

Reads a 32-bit SDO in expedited mode.

See **ec_slave_sdo_read_exp()**(p. 67)

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
sdo_index SDO index
sdo_subindex SDO subindex
target memory for read value

Definition at line 645 of file canopen.c.

8.1.2.5 int ecrt_slave_sdo_write_exp8 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint8_t *value*)

Writes an 8-bit SDO in expedited mode.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

value new value

Definition at line 665 of file canopen.c.

8.1.2.6 int ecrt_slave_sdo_write_exp16 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint16_t *value*)

Writes a 16-bit SDO in expedited mode.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

value new value

Definition at line 682 of file canopen.c.

8.1.2.7 int ecrt_slave_sdo_write_exp32 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint32_t *value*)

Writes a 32-bit SDO in expedited mode.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

value new value

Definition at line 701 of file canopen.c.

8.1.2.8 ec_slave_t* ecrt_domain_register_field (ec_domain_t * *domain*, const char * *address*, const char * *vendor_name*, const char * *product_name*, void ** *data_ptr*, const char * *field_name*, unsigned int *field_index*, unsigned int *field_count*)

Registers a data field in a domain.

- If *data_ptr* is NULL, the slave is only checked against its type.
- If *field_count* is 0, it is assumed that one data field is to be registered.
- If *field_count* is greater then 1, it is assumed that *data_ptr* is an array of the respective size.

Returns:
pointer to the slave on success, else NULL

Parameters:

domain EtherCAT domain

address ASCII address of the slave, see **ecrt_master_get_slave()**(p. 23)

vendor_name vendor name

product_name product name

data_ptr address of the process data pointer

field_name data field name

field_index offset of data fields with *field_type*

field_count number of data fields (with the same type) to register

Definition at line 370 of file domain.c.

8.1.2.9 int ecrt_domain_register_field_list (ec_domain_t * *domain*, const ec_field_init_t * *fields*)

Registeres a bunch of data fields.

Caution! The list has to be terminated with a NULL structure ({})!

Returns:
0 in case of success, else < 0

Parameters:

domain EtherCAT domain

fields array of data field registrations

Definition at line 462 of file domain.c.

8.1.2.10 void ecrt_domain_queue (ec_domain_t * *domain*)

Places all process data datagrams in the masters datagram queue.

Parameters:

domain EtherCAT domain

Definition at line 488 of file domain.c.

8.1.2.11 void ecrt_domain_process (ec_domain_t * *domain*)

Processes received process data.

Parameters:
domain EtherCAT domain

Definition at line 504 of file domain.c.

8.1.2.12 int ecrt_domain_state (ec_domain_t * *domain*)

Returns the state of a domain.

Returns:
0 if all datagrams were received, else -1.

Parameters:
domain EtherCAT domain

Definition at line 528 of file domain.c.

8.1.2.13 ec_domain_t* ecrt_master_create_domain (ec_master_t * *master*)

Creates a domain.

Returns:
pointer to new domain on success, else NULL

Parameters:
master master

Definition at line 1070 of file master.c.

8.1.2.14 int ecrt_master_activate (ec_master_t * *master*)

Configures all slaves and leads them to the OP state.

Does the complete configuration and activation for all slaves. Sets sync managers and FMMUs, and does the appropriate transitions, until the slave is operational.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

Definition at line 1116 of file master.c.

8.1.2.15 void ecrt_master_deactivate (ec_master_t * *master*)

Resets all slaves to INIT state.

Parameters:
master EtherCAT master

Definition at line 1294 of file master.c.

8.1.2.16 int ecrt_master_fetch_sdo_lists (ec_master_t * *master*)

Fetches the SDO dictionaries of all slaves.

Slaves that do not support the CoE protocol are left out.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

Definition at line 1314 of file master.c.

8.1.2.17 void ecrt_master_sync_io (ec_master_t * *master*)

Sends queued datagrams and waits for their reception.

Parameters:
master EtherCAT master

Definition at line 1338 of file master.c.

8.1.2.18 void ecrt_master_async_send (ec_master_t * *master*)

Asynchronous sending of datagrams.

Parameters:
master EtherCAT master

Definition at line 1394 of file master.c.

8.1.2.19 void ecrt_master_async_receive (ec_master_t * *master*)

Asynchronous receiving of datagrams.

Parameters:
master EtherCAT master

Definition at line 1421 of file master.c.

8.1.2.20 void ecrt_master_prepare_async_io (ec_master_t * *master*)

Prepares synchronous IO.

Queues all domain datagrams and sends them. Then waits a certain time, so that ecrt_master_sasync_-receive() can be called securely.

Parameters:
master EtherCAT master

Definition at line 1463 of file master.c.

8.1.2.21 void ecrt_master_run (ec_master_t * *master*)

Does all cyclic master work.

Parameters:
master EtherCAT master

Definition at line 1491 of file master.c.

8.1.2.22 ec_slave_t* ecrt_master_get_slave (const ec_master_t * *master*, const char * *address*)

Translates an ASCII coded bus-address to a slave pointer.

These are the valid addressing schemes:

- *"X"* = the X. slave on the bus,
- *"X:Y"* = the Y. slave after the X. branch (bus coupler),
- *"#X"* = the slave with alias X,
- *"#X:Y"* = the Y. slave after the branch (bus coupler) with alias X. X and Y are zero-based indices and may be provided in hexadecimal or octal notation (with respective prefix).

Returns:
pointer to the slave on success, else NULL

Parameters:
master Master

address address string

Definition at line 1515 of file master.c.

8.1.2.23 void ecrt_master_callbacks (ec_master_t * *master*, int(*)(void *) *request_cb*, void(*)(void *) *release_cb*, void * *cb_data*)

Sets the locking callbacks.

The request_cb function must return zero, to allow another instance (the EoE process for example) to access the master. Non-zero means, that access is forbidden at this time.

Parameters:
master EtherCAT master
request_cb request lock CB
release_cb release lock CB
cb_data data parameter

Definition at line 1618 of file master.c.

8.1.2.24 int ecrt_master_start_eoe (ec_master_t * *master*)

Starts Ethernet-over-EtherCAT processing for all EoE-capable slaves.

Parameters:
master EtherCAT master

Definition at line 1636 of file master.c.

8.1.2.25 void ecrt_master_debug (ec_master_t * *master*, int *level*)

Sets the debug level of the master.

The following levels are valid:

- 1: only output positions marks and basic data
- 2: additional frame data output

Parameters:
master EtherCAT master
level debug level

Definition at line 1657 of file master.c.

8.1.2.26 void ecrt_master_print (const ec_master_t * *master*, unsigned int *verbosity*)

Outputs all master information.

Verbosity:

- 0: Only slave types and positions
- 1: with EEPROM contents
- >1: with SDO dictionaries

Parameters:
master EtherCAT master
verbosity verbosity level

Definition at line 1678 of file master.c.

8.1.2.27 ec_master_t* ecrt_request_master (unsigned int *master_index*)

Reserves an EtherCAT master for realtime operation.

Returns:
pointer to reserved master, or NULL on error

Parameters:
master_index master index

Definition at line 401 of file module.c.

8.1.2.28 void ecrt_release_master (ec_master_t * *master*)

Releases a reserved EtherCAT master.

Parameters:
master EtherCAT master

Definition at line 459 of file module.c.

8.1.2.29 int ecrt_slave_write_alias (ec_slave_t * *slave*, uint16_t *alias*)

Writes the "configured station alias" to the slave's EEPROM.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
alias new alias

Definition at line 1540 of file slave.c.

8.1.2.30 int ecrt_slave_field_size (ec_slave_t * *slave*, const char * *field_name*, unsigned int *field_index*, size_t *size*)

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
field_name data field name
field_index data field index
size new data field size

Definition at line 1554 of file slave.c.

8.2 EtherCAT device interface

8.2.1 Detailed Description

Master interface for EtherCAT-capable network device drivers.

Through the EtherCAT device interface, EtherCAT-capable network device drivers are able to connect their device(s) to the master, pass received frames and notify the master about status changes. The master on his part, can send his frames through connected devices.

Functions

- void **ecdev_receive** (**ec_device_t** ∗device, const void ∗data, size_t size)

  Accepts a received frame.

- void **ecdev_link_state** (**ec_device_t** ∗device, uint8_t state)

  Sets a new link state.

- **ec_device_t** ∗ **ecdev_register** (unsigned int master_index, struct net_device ∗net_dev, **ec_isr_t** isr, struct module ∗module)

  Connects an EtherCAT device to a certain master.

- void **ecdev_unregister** (unsigned int master_index, **ec_device_t** ∗device)

  Disconnect an EtherCAT device from the master.

- int **ecdev_start** (unsigned int master_index)

  Starts the master associated with the device.

- void **ecdev_stop** (unsigned int master_index)

  Stops the master associated with the device.

8.2.2 Function Documentation

8.2.2.1 void ecdev_receive (ec_device_t ∗ *device*, const void ∗ *data*, size_t *size*)

Accepts a received frame.

Forwards the received data to the master. The master will analyze the frame and dispatch the received commands to the sending instances.

Parameters:

device EtherCAT device

data pointer to received data

size number of bytes received

Definition at line 235 of file device.c.

8.2.2.2 void ecdev_link_state (ec_device_t * *device*, uint8_t *state*)

Sets a new link state.

If the device notifies the master about the link being down, the master will not try to send frames using this device.

Parameters:

device EtherCAT device

state new link state

Definition at line 260 of file device.c.

8.2.2.3 ec_device_t* ecdev_register (unsigned int *master_index*, struct net_device * *net_dev*, ec_isr_t *isr*, struct module * *module*)

Connects an EtherCAT device to a certain master.

The master will use the device for sending and receiving frames. It is required that no other instance (for example the kernel IP stack) uses the device.

Returns:

0 on success, else < 0

Parameters:

master_index master index

net_dev net_device of the device

isr interrupt service routine

module pointer to the module

Definition at line 282 of file module.c.

8.2.2.4 void ecdev_unregister (unsigned int *master_index*, ec_device_t * *device*)

Disconnect an EtherCAT device from the master.

The device is disconnected from the master and all device ressources are freed.

Attention:

Before calling this function, the **ecdev_stop**()(p. 28) function has to be called, to be sure that the master does not use the device any more.

Parameters:

master_index master index

device EtherCAT device

Definition at line 329 of file module.c.

8.2.2.5 int ecdev_start (unsigned int *master_index*)

Starts the master associated with the device.

This function has to be called by the network device driver to tell the master that the device is ready to send and receive data. The master will enter the idle mode then.

Parameters:
master_index master index

Definition at line 357 of file module.c.

8.2.2.6 void ecdev_stop (unsigned int *master_index*)

Stops the master associated with the device.

Tells the master to stop using the device for frame IO. Has to be called before unregistering the device.

Parameters:
master_index master index

Definition at line 380 of file module.c.

Chapter 9

IgH EtherCAT master Directory Documentation

9.1 devices/ Directory Reference

Files

- file **ecdev.h**

  EtherCAT interface for EtherCAT device drivers.

9.2 include/ Directory Reference

Files

- file **ecrt.h**

  EtherCAT realtime interface.

9.3 master/ Directory Reference

Files

- file **canopen.c**

  Canopen-over-EtherCAT functions.

- file **datagram.c**

  Methods of an EtherCAT datagram.

- file **datagram.h**

  EtherCAT datagram structure.

- file **debug.c**

  Ethernet interface for debugging purposes.

- file **debug.h**

  Network interface for debugging purposes.

- file **device.c**

  EtherCAT device methods.

- file **device.h**

  EtherCAT device structure.

- file **domain.c**

  EtherCAT domain methods.

- file **domain.h**

  EtherCAT domain structure.

- file **doxygen.c**
- file **ethernet.c**

  Ethernet-over-EtherCAT (EoE).

- file **ethernet.h**

  Ethernet-over-EtherCAT (EoE).

- file **fsm.c**

  EtherCAT finite state machines.

- file **fsm.h**

  EtherCAT finite state machines.

- file **globals.h**

  Global definitions and macros.

- file **mailbox.c**

  Mailbox functionality.

- file **mailbox.h**

  Mailbox functionality.

- file **master.c**

  EtherCAT master methods.

- file **master.h**

  EtherCAT master structure.

- file **module.c**

  EtherCAT master driver module.

- file **slave.c**

  EtherCAT slave methods.

- file **slave.h**

  EtherCAT stave structure.

- file **types.c**

  EtherCAT slave descriptions.

- file **types.h**

  EtherCAT slave types.

Chapter 10

IgH EtherCAT master Data Structure Documentation

10.1 ec_address_t Union Reference

10.1.1 Detailed Description

EtherCAT address.

Definition at line 90 of file datagram.h.

Data Fields

- struct {
  uint16_t **slave**
  configured or autoincrement address
  uint16_t **mem**
  physical memory address
  } **physical**

  physical address

- uint32_t **logical**
  logical address

10.2 ec_code_msg_t Struct Reference

10.2.1 Detailed Description

Code - Message pair.

Some EtherCAT datagrams support reading a status code to display a certain message. This type allows to map a code to a message string.

Definition at line 185 of file globals.h.

Data Fields

- uint32_t **code**

  code

- const char * **message**

  message belonging to code

10.3 ec_datagram_t Struct Reference

10.3.1 Detailed Description

EtherCAT datagram.

Definition at line 109 of file datagram.h.

Data Fields

- list_head **list**
  needed by domain datagram lists
- list_head **queue**
  master datagram queue item
- **ec_datagram_type_t type**
  datagram type (APRD, BWR, etc)
- **ec_address_t address**
  receipient address
- uint8_t * **data**
  datagram data
- size_t **mem_size**
  datagram data *memory size*
- size_t **data_size**
  size of the data in data
- uint8_t **index**
  datagram index (set by master)
- uint16_t **working_counter**
  working counter
- **ec_datagram_state_t state**
  datagram state
- cycles_t **t_sent**
  time, the datagrams was sent

10.4 ec_debug_t Struct Reference

10.4.1 Detailed Description

Debugging network interface.

Definition at line 49 of file debug.h.

Data Fields

- net_device * **dev**
  net_device for virtual ethernet device
- net_device_stats **stats**
  device statistics
- uint8_t **opened**
  net_device is opened

10.5 ec_device Struct Reference

10.5.1 Detailed Description

EtherCAT device.

An EtherCAT device is a network interface card, that is owned by an EtherCAT master to send and receive EtherCAT frames with.

Definition at line 59 of file device.h.

Data Fields

- **ec_master_t * master**
  EtherCAT master.
- net_device * **dev**
  pointer to the assigned net_device
- uint8_t **open**
  true, if the net_device has been opened
- sk_buff * **tx_skb**
  transmit socket buffer
- **ec_isr_t isr**
  pointer to the device's interrupt service routine
- **module * module**
  pointer to the device's owning module
- uint8_t **link_state**
  device link state
- **ec_debug_t dbg**
  debug device

10.6 ec_domain Struct Reference

10.6.1 Detailed Description

EtherCAT domain.

Handles the process data and the therefore needed datagrams of a certain group of slaves.

Definition at line 75 of file domain.h.

Data Fields

- kobject **kobj**
  kobject
- list_head **list**
  list item
- unsigned int **index**
  domain index (just a number)
- **ec_master_t** * **master**
  EtherCAT master owning the domain.
- size_t **data_size**
  size of the process data
- list_head **datagrams**
  process data datagrams
- uint32_t **base_address**
  logical offset address of the process data
- unsigned int **response_count**
  number of responding slaves
- list_head **field_regs**
  data field registrations

10.7 ec_eeprom_pdo_entry_t Struct Reference

10.7.1 Detailed Description

PDO entry description (EEPROM).

Definition at line 171 of file slave.h.

Data Fields

- list_head **list**
  list item
- uint16_t **index**
  PDO index.
- uint8_t **subindex**
  entry subindex
- char ∗ **name**
  entry name
- uint8_t **bit_length**
  entry length in bit

10.8 ec_eeprom_pdo_t Struct Reference

10.8.1 Detailed Description

PDO description (EEPROM).

Definition at line 154 of file slave.h.

Data Fields

- list_head **list**
  list item
- **ec_pdo_type_t type**
  PDO type.
- uint16_t **index**
  PDO index.
- uint8_t **sync_manager**
  assigned sync manager
- char * **name**
  PDO name.
- list_head **entries**
  entry list

10.9 ec_eeprom_string_t Struct Reference

10.9.1 Detailed Description

String object (EEPROM).

Definition at line 110 of file slave.h.

Data Fields

- list_head **list**
  list item
- size_t **size**
  size in bytes
- char * **data**
  string data

10.10 ec_eeprom_sync_t Struct Reference

10.10.1 Detailed Description

Sync manager configuration (EEPROM).

Definition at line 124 of file slave.h.

Data Fields

- list_head **list**
  list item
- unsigned int **index**
  sync manager index
- uint16_t **physical_start_address**
  physical start address
- uint16_t **length**
  data length in bytes
- uint8_t **control_register**
  control register value
- uint8_t **enable**
  enable bit

10.11 ec_eoe Struct Reference

10.11.1 Detailed Description

Ethernet-over-EtherCAT (EoE) handler.

The master creates one of these objects for each slave that supports the EoE protocol.

Definition at line 72 of file ethernet.h.

Data Fields

- list_head **list**
  list item
- **ec_slave_t** ∗ **slave**
  pointer to the corresponding slave
- **ec_datagram_t datagram**
  datagram
- void(∗ **state**)(**ec_eoe_t** ∗)
  state function for the state machine
- net_device ∗ **dev**
  net_device for virtual ethernet device
- net_device_stats **stats**
  device statistics
- unsigned int **opened**
  net_device is opened
- sk_buff ∗ **rx_skb**
  current rx socket buffer
- off_t **rx_skb_offset**
  current write pointer in the socket buffer
- size_t **rx_skb_size**
  size of the allocated socket buffer memory
- uint8_t **rx_expected_fragment**
  next expected fragment number
- list_head **tx_queue**
  queue for frames to send
- unsigned int **tx_queue_active**
  kernel netif queue started

- unsigned int **tx_queued_frames**
  number of frames in the queue
- spinlock_t **tx_queue_lock**
  spinlock for the send queue
- **ec_eoe_frame_t** * **tx_frame**
  current TX frame
- uint8_t **tx_frame_number**
  number of the transmitted frame
- uint8_t **tx_fragment_number**
  number of the fragment
- size_t **tx_offset**
  number of octets sent

10.12 ec_eoe_frame_t Struct Reference

10.12.1 Detailed Description

Queued frame structure.

Definition at line 55 of file ethernet.h.

Data Fields

- list_head **queue**
  list item
- sk_buff * **skb**
  socket buffer

10.13 ec_field_init_t Struct Reference

10.13.1 Detailed Description

Initialization type for field registrations.

This type is used as a parameter for the ec_domain_register_field_list() function.

Definition at line 77 of file ecrt.h.

Data Fields

- void ∗∗ **data_ptr**
  address of the process data pointer
- const char ∗ **slave_address**
  slave address string (see **ecrt_master_get_slave()**(p. 23)*)*
- const char ∗ **vendor_name**
  vendor name
- const char ∗ **product_name**
  product name
- const char ∗ **field_name**
  data field name
- unsigned int **field_index**
  index in data fields with same name
- unsigned int **field_count**
  number of data fields with same name

10.14 ec_field_reg_t Struct Reference

10.14.1 Detailed Description

Data field registration type.

Definition at line 57 of file domain.h.

Data Fields

- list_head **list**
  list item
- **ec_slave_t** * **slave**
  slave
- const **ec_sync_t** * **sync**
  sync manager
- uint32_t **field_offset**
  data field offset
- void ** **data_ptr**
  pointer to process data pointer(s)

10.15 ec_field_t Struct Reference

10.15.1 Detailed Description

Process data field.

Definition at line 74 of file types.h.

Data Fields

- const char * **name**
  field name
- size_t **size**
  field size in bytes

10.16 ec_fmmu_t Struct Reference

10.16.1 Detailed Description

FMMU configuration.

Definition at line 96 of file slave.h.

Data Fields

- const **ec_domain_t** ∗ **domain**
  domain
- const **ec_sync_t** ∗ **sync**
  sync manager
- uint32_t **logical_start_address**
  logical start address

10.17 ec_fsm Struct Reference

10.17.1 Detailed Description

Finite state machine of an EtherCAT master.

Definition at line 56 of file fsm.h.

Data Fields

- **ec_master_t * master**
  master the FSM runs on
- **ec_slave_t * slave**
  slave the FSM runs on
- **ec_datagram_t datagram**
  datagram used in the state machine
- void(* **master_state**)(**ec_fsm_t** *)
  master state function
- unsigned int **master_slaves_responding**
  number of responding slaves
- **ec_slave_state_t master_slave_states**
  states of responding slaves
- unsigned int **master_validation**
  non-zero, if validation to do
- void(* **slave_state**)(**ec_fsm_t** *)
  slave state function
- void(* **sii_state**)(**ec_fsm_t** *)
  SII state function.
- uint16_t **sii_offset**
  input: offset in SII
- unsigned int **sii_mode**
  SII reading done by APRD (0) or NPRD (1).
- uint8_t **sii_value** [4]
  raw SII value (32bit)
- cycles_t **sii_start**
  sii start
- void(* **change_state**)(**ec_fsm_t** *)

slave state change state function

- uint8_t **change_new**

  input: new state

- cycles_t **change_start**

  change start

10.18 ec_master Struct Reference

10.18.1 Detailed Description

EtherCAT master.

Manages slaves, domains and IO.

Definition at line 89 of file master.h.

Data Fields

- list_head **list**
  list item for module's master list
- unsigned int **reserved**
  non-zero, if the master is reserved for RT
- unsigned int **index**
  master index
- kobject **kobj**
  kobject
- list_head **slaves**
  list of slaves on the bus
- unsigned int **slave_count**
  number of slaves on the bus
- **ec_device_t** * **device**
  EtherCAT device.
- list_head **datagram_queue**
  datagram queue
- uint8_t **datagram_index**
  current datagram index
- list_head **domains**
  list of domains
- **ec_datagram_t simple_datagram**
  datagram structure for initialization
- unsigned int **timeout**
  timeout in synchronous IO
- int **debug_level**
  master debug level

- **ec_stats_t stats**
  cyclic statistics
- workqueue_struct * **workqueue**
  master workqueue
- work_struct **idle_work**
  free run work object
- **ec_fsm_t fsm**
  master state machine
- **ec_master_mode_t mode**
  master mode
- timer_list **eoe_timer**
  EoE timer object.
- unsigned int **eoe_running**
  non-zero, if EoE processing is active.
- list_head **eoe_handlers**
  Ethernet-over-EtherCAT handlers.
- spinlock_t **internal_lock**
  spinlock used in idle mode
- int(* **request_cb**)(void *)
  lock request callback
- void(* **release_cb**)(void *)
  lock release callback
- void * **cb_data**
  data parameter of locking callbacks
- uint8_t **eeprom_write_enable**
  allow write operations to EEPROMs

10.19 ec_sdo_entry_t Struct Reference

10.19.1 Detailed Description

CANopen SDO entry.

Definition at line 204 of file slave.h.

Data Fields

- list_head **list**
  list item
- uint8_t **subindex**
  entry subindex
- uint16_t **data_type**
  entry data type
- uint16_t **bit_length**
  entry length in bit
- char * **name**
  entry name

10.20 ec_sdo_t Struct Reference

10.20.1 Detailed Description

CANopen SDO.

Definition at line 187 of file slave.h.

Data Fields

- list_head **list**
  list item
- uint16_t **index**
  SDO index.
- uint8_t **object_code**
  object code
- char * **name**
  SDO name.
- list_head **entries**
  entry list

10.21 ec_slave Struct Reference

10.21.1 Detailed Description

EtherCAT slave.

Definition at line 234 of file slave.h.

Data Fields

- list_head **list**
  list item
- kobject **kobj**
  kobject
- **ec_master_t** * **master**
  master owning the slave
- uint16_t **ring_position**
  ring position
- uint16_t **station_address**
  configured station address
- uint16_t **coupler_index**
  index of the last bus coupler
- uint16_t **coupler_subindex**
  index of this slave after last coupler
- uint8_t **base_type**
  slave type
- uint8_t **base_revision**
  revision
- uint16_t **base_build**
  build number
- uint16_t **base_fmmu_count**
  number of supported FMMUs
- uint16_t **base_sync_count**
  number of supported sync managers
- uint8_t **dl_link** [4]
  link detected
- uint8_t **dl_loop** [4]

loop closed

- uint8_t **dl_signal** [4]
  detected signal on RX port
- uint16_t **sii_alias**
  configured station alias
- uint32_t **sii_vendor_id**
  vendor id
- uint32_t **sii_product_code**
  vendor's product code
- uint32_t **sii_revision_number**
  revision number
- uint32_t **sii_serial_number**
  serial number
- uint16_t **sii_rx_mailbox_offset**
  mailbox address (master to slave)
- uint16_t **sii_rx_mailbox_size**
  mailbox size (master to slave)
- uint16_t **sii_tx_mailbox_offset**
  mailbox address (slave to master)
- uint16_t **sii_tx_mailbox_size**
  mailbox size (slave to master)
- uint16_t **sii_mailbox_protocols**
  supported mailbox protocols
- uint8_t **sii_physical_layer** [4]
  port media
- const **ec_slave_type_t** ∗ **type**
  pointer to slave type object
- uint8_t **registered**
  true, if slave has been registered
- **ec_fmmu_t fmmus** [EC_MAX_FMMUS]
  FMMU configurations.
- uint8_t **fmmu_count**
  number of FMMUs used

- uint8_t * **eeprom_data**
  Complete EEPROM image.
- uint16_t **eeprom_size**
  size of the EEPROM contents in byte
- list_head **eeprom_strings**
  EEPROM STRING categories.
- list_head **eeprom_syncs**
  EEPROM SYNC MANAGER categories.
- list_head **eeprom_pdos**
  EEPROM [RT]XPDO categories.
- char * **eeprom_group**
  slave group acc.
- char * **eeprom_image**
  slave image name acc.
- char * **eeprom_order**
  slave order number acc.
- char * **eeprom_name**
  slave name acc.
- uint16_t * **new_eeprom_data**
  new EEPROM data to write
- size_t **new_eeprom_size**
  size of new EEPROM data in words
- list_head **sdo_dictionary**
  SDO directory list.
- **ec_slave_state_t requested_state**
  requested slave state
- **ec_slave_state_t current_state**
  current slave state
- unsigned int **error_flag**
  stop processing after an error
- unsigned int **online**
  non-zero, if the slave responds.
- list_head **varsize_fields**
  size information for variable-sized data fields.

10.21.2 Field Documentation

10.21.2.1 char* ec_slave::eeprom_group

slave group acc.

to EEPROM

Definition at line 283 of file slave.h.

10.21.2.2 char* ec_slave::eeprom_image

slave image name acc.

to EEPROM

Definition at line 284 of file slave.h.

10.21.2.3 char* ec_slave::eeprom_order

slave order number acc.

to EEPROM

Definition at line 285 of file slave.h.

10.21.2.4 char* ec_slave::eeprom_name

slave name acc.

to EEPROM

Definition at line 286 of file slave.h.

10.22 ec_slave_ident_t Struct Reference

10.22.1 Detailed Description

Slave type identification.

Definition at line 118 of file types.h.

Data Fields

- uint32_t **vendor_id**
  vendor id
- uint32_t **product_code**
  product code
- const **ec_slave_type_t** * **type**
  associated slave description object

10.23 ec_slave_type Struct Reference

10.23.1 Detailed Description

Slave description type.

Definition at line 102 of file types.h.

Data Fields

- const char * **vendor_name**
  vendor name
- const char * **product_name**
  product name
- const char * **description**
  free description
- **ec_special_type_t special**
  special slave type?
- const **ec_sync_t** * **sync_managers** [EC_MAX_SYNC]
  sync managers

10.24 ec_stats_t Struct Reference

10.24.1 Detailed Description

Cyclic statistics.

Definition at line 72 of file master.h.

Data Fields

- unsigned int **timeouts**
  datagram timeouts
- unsigned int **delayed**
  delayed datagrams
- unsigned int **corrupted**
  corrupted frames
- unsigned int **unmatched**
  unmatched datagrams
- cycles_t **t_last**
  time of last output

10.25 ec_sync_t Struct Reference

10.25.1 Detailed Description

Sync manager.

Definition at line 87 of file types.h.

Data Fields

- uint16_t **physical_start_address**
  physical start address
- uint16_t **size**
  size in bytes
- uint8_t **control_byte**
  control register value
- const **ec_field_t** * **fields** [EC_MAX_FIELDS]
  field array

10.26 ec_varsize_t Struct Reference

10.26.1 Detailed Description

Variable-sized field information.

Definition at line 220 of file slave.h.

Data Fields

- list_head **list**
  list item
- const **ec_field_t** * **field**
  data field
- size_t **size**
  field size

Chapter 11

IgH EtherCAT master File Documentation

11.1 canopen.c File Reference

11.1.1 Detailed Description

Canopen-over-EtherCAT functions.

Definition in file **canopen.c**.

Functions

- void **ec_canopen_abort_msg** (uint32_t abort_code)

  Outputs an SDO abort message.

- int **ec_slave_fetch_sdo_descriptions** (**ec_slave_t** ∗slave,**ec_datagram_t** ∗datagram)

  Fetches the SDO descriptions for the known SDOs.

- int **ec_slave_fetch_sdo_entries** (**ec_slave_t** ∗slave,**ec_datagram_t** ∗datagram,**ec_sdo_t** ∗sdo,uint8_t subindices)

  Fetches all entries (subindices) to an SDO.

- int **ec_slave_sdo_read_exp** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t ∗target)

  Reads 32 bit of a CANopen SDO in expedited mode.

- int **ec_slave_sdo_write_exp** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, const uint8_t ∗sdo_data, size_t size)

  Writes a CANopen SDO using expedited mode.

- int **ecrt_slave_sdo_read** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_-t ∗target, size_t ∗size)

  Reads a CANopen SDO in normal mode.

- int **ec_slave_fetch_sdo_list** (**ec_slave_t** ∗slave)

  Fetches the SDO dictionary of a slave.

- int **ecrt_slave_sdo_read_exp8** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t ∗target)

  Reads an 8-bit SDO in expedited mode.

- int **ecrt_slave_sdo_read_exp16** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t ∗target)

  Reads a 16-bit SDO in expedited mode.

- int **ecrt_slave_sdo_read_exp32** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t ∗target)

  Reads a 32-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp8** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t value)

  Writes an 8-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp16** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t value)

  Writes a 16-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp32** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t value)

  Writes a 32-bit SDO in expedited mode.

Variables

- const **ec_code_msg_t sdo_abort_messages** [ ]

  SDO abort messages.

11.1.2 Function Documentation

11.1.2.1 int ec_slave_fetch_sdo_descriptions (ec_slave_t ∗ *slave*, ec_datagram_t ∗ *datagram*)

Fetches the SDO descriptions for the known SDOs.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

datagram datagram

Definition at line 361 of file canopen.c.

11.1.2.2 int ec_slave_fetch_sdo_entries (ec_slave_t * *slave*, ec_datagram_t * *datagram*, ec_sdo_t * *sdo*, uint8_t *subindices*)

Fetches all entries (subindices) to an SDO.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

datagram datagram

sdo SDO

subindices number of subindices

Definition at line 446 of file canopen.c.

11.1.2.3 int ec_slave_sdo_read_exp (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint8_t * *target*)

Reads 32 bit of a CANopen SDO in expedited mode.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

target 4-byte memory

Definition at line 62 of file canopen.c.

11.1.2.4 int ec_slave_sdo_write_exp (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, const uint8_t * *sdo_data*, size_t *size*)

Writes a CANopen SDO using expedited mode.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

sdo_data new value

size Data size in bytes (1 - 4)

Definition at line 121 of file canopen.c.

11.1.2.5 int ec_slave_fetch_sdo_list (ec_slave_t * *slave*)

Fetches the SDO dictionary of a slave.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

Definition at line 270 of file canopen.c.

11.1.3 Variable Documentation

11.1.3.1 const ec_code_msg_t sdo_abort_messages[ ]

SDO abort messages.

The "abort SDO transfer request" supplies an abort code, which can be translated to clear text. This table does the mapping of the codes and messages.

Definition at line 534 of file canopen.c.

11.2 datagram.c File Reference

11.2.1 Detailed Description

Methods of an EtherCAT datagram.

Definition in file **datagram.c**.

Functions

- void **ec_datagram_init** (**ec_datagram_t** ∗datagram)
  Datagram constructor.
- void **ec_datagram_clear** (**ec_datagram_t** ∗datagram)
  Datagram destructor.
- int **ec_datagram_prealloc** (**ec_datagram_t** ∗datagram, size_t size)
  Allocates datagram data memory.
- int **ec_datagram_nprd** (**ec_datagram_t** ∗datagram, uint16_t node_address, uint16_t offset, size_t data_size)
  Initializes an EtherCAT NPRD datagram.
- int **ec_datagram_npwr** (**ec_datagram_t** ∗datagram, uint16_t node_address, uint16_t offset, size_t data_size)
  Initializes an EtherCAT NPWR datagram.
- int **ec_datagram_aprd** (**ec_datagram_t** ∗datagram, uint16_t ring_position, uint16_t offset, size_t data_size)
  Initializes an EtherCAT APRD datagram.
- int **ec_datagram_apwr** (**ec_datagram_t** ∗datagram, uint16_t ring_position, uint16_t offset, size_t data_size)
  Initializes an EtherCAT APWR datagram.
- int **ec_datagram_brd** (**ec_datagram_t** ∗datagram, uint16_t offset, size_t data_size)
  Initializes an EtherCAT BRD datagram.
- int **ec_datagram_bwr** (**ec_datagram_t** ∗datagram, uint16_t offset, size_t data_size)
  Initializes an EtherCAT BWR datagram.
- int **ec_datagram_lrw** (**ec_datagram_t** ∗datagram, uint32_t offset, size_t data_size)
  Initializes an EtherCAT LRW datagram.

11.2.2 Function Documentation

11.2.2.1 void ec_datagram_init (ec_datagram_t ∗ *datagram*)

Datagram constructor.

Parameters:
datagram EtherCAT datagram

Definition at line 70 of file datagram.c.

11.2.2.2 void ec_datagram_clear (ec_datagram_t * *datagram*)

Datagram destructor.

Parameters:
datagram EtherCAT datagram

Definition at line 89 of file datagram.c.

11.2.2.3 int ec_datagram_prealloc (ec_datagram_t * *datagram*, size_t *size*)

Allocates datagram data memory.

If the allocated memory is already larger than requested, nothing ist done.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram

size New size in bytes

Definition at line 102 of file datagram.c.

11.2.2.4 int ec_datagram_nprd (ec_datagram_t * *datagram*, uint16_t *node_address*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT NPRD datagram.

Node-adressed physical read.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram

node_address configured station address

offset physical memory address

data_size number of bytes to read

Definition at line 131 of file datagram.c.

11.2.2.5 int ec_datagram_npwr (ec_datagram_t * *datagram*, uint16_t *node_address*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT NPWR datagram.

Node-adressed physical write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
node_address configured station address
offset physical memory address
data_size number of bytes to write

Definition at line 159 of file datagram.c.

11.2.2.6 int ec_datagram_aprd (ec_datagram_t * *datagram*, uint16_t *ring_position*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT APRD datagram.

Autoincrement physical read.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
ring_position auto-increment position
offset physical memory address
data_size number of bytes to read

Definition at line 187 of file datagram.c.

11.2.2.7 int ec_datagram_apwr (ec_datagram_t * *datagram*, uint16_t *ring_position*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT APWR datagram.

Autoincrement physical write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
ring_position auto-increment position
offset physical memory address
data_size number of bytes to write

Definition at line 212 of file datagram.c.

11.2.2.8 int ec_datagram_brd (ec_datagram_t * *datagram*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT BRD datagram.

Broadcast read.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
offset physical memory address
data_size number of bytes to read

Definition at line 237 of file datagram.c.

11.2.2.9 int ec_datagram_bwr (ec_datagram_t * *datagram*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT BWR datagram.

Broadcast write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
offset physical memory address
data_size number of bytes to write

Definition at line 260 of file datagram.c.

11.2.2.10 int ec_datagram_lrw (ec_datagram_t * *datagram*, uint32_t *offset*, size_t *data_size*)

Initializes an EtherCAT LRW datagram.

Logical read write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
offset logical address
data_size number of bytes to read/write

Definition at line 283 of file datagram.c.

11.3 datagram.h File Reference

11.3.1 Detailed Description

EtherCAT datagram structure.

Definition in file **datagram.h**.

Data Structures

- union **ec_address_t**

  EtherCAT address.

- struct **ec_datagram_t**

  EtherCAT datagram.

Enumerations

- enum **ec_datagram_type_t** {

  EC_CMD_NONE = 0x00, **EC_CMD_APRD** = 0x01, **EC_CMD_APWR** = 0x02, **EC_CMD_NPRD** = 0x04,

  EC_CMD_NPWR = 0x05, **EC_CMD_BRD** = 0x07, **EC_CMD_BWR** = 0x08, **EC_CMD_LRW** = 0x0C }

  EtherCAT datagram type.

- enum **ec_datagram_state_t** {

  EC_CMD_INIT, **EC_CMD_QUEUED**, **EC_CMD_SENT**, **EC_CMD_RECEIVED**,

  EC_CMD_TIMEOUT, **EC_CMD_ERROR** }

  EtherCAT datagram state.

Functions

- void **ec_datagram_init** (**ec_datagram_t** ∗)

  Datagram constructor.

- void **ec_datagram_clear** (**ec_datagram_t** ∗)

  Datagram destructor.

- int **ec_datagram_prealloc** (**ec_datagram_t** ∗, size_t)

  Allocates datagram data memory.

- int **ec_datagram_nprd** (**ec_datagram_t** ∗, uint16_t, uint16_t, size_t)

  Initializes an EtherCAT NPRD datagram.

- int **ec_datagram_npwr** (**ec_datagram_t** ∗, uint16_t, uint16_t, size_t)

  Initializes an EtherCAT NPWR datagram.

- int **ec_datagram_aprd** (**ec_datagram_t** *, uint16_t, uint16_t, size_t)
  Initializes an EtherCAT APRD datagram.
- int **ec_datagram_apwr** (**ec_datagram_t** *, uint16_t, uint16_t, size_t)
  Initializes an EtherCAT APWR datagram.
- int **ec_datagram_brd** (**ec_datagram_t** *, uint16_t, size_t)
  Initializes an EtherCAT BRD datagram.
- int **ec_datagram_bwr** (**ec_datagram_t** *, uint16_t, size_t)
  Initializes an EtherCAT BWR datagram.
- int **ec_datagram_lrw** (**ec_datagram_t** *, uint32_t, size_t)
  Initializes an EtherCAT LRW datagram.

11.3.2 Enumeration Type Documentation

11.3.2.1 enum ec_datagram_type_t

EtherCAT datagram type.

Enumerator:

EC_CMD_NONE Dummy.
EC_CMD_APRD Auto-increment physical read.
EC_CMD_APWR Auto-increment physical write.
EC_CMD_NPRD Node-addressed physical read.
EC_CMD_NPWR Node-addressed physical write.
EC_CMD_BRD Broadcast read.
EC_CMD_BWR Broadcast write.
EC_CMD_LRW Logical read/write.

Definition at line 56 of file datagram.h.

11.3.2.2 enum ec_datagram_state_t

EtherCAT datagram state.

Enumerator:

EC_CMD_INIT new datagram
EC_CMD_QUEUED datagram queued by master
EC_CMD_SENT datagram has been sent
EC_CMD_RECEIVED datagram has been received
EC_CMD_TIMEOUT datagram timed out
EC_CMD_ERROR error while sending/receiving

Definition at line 73 of file datagram.h.

11.3.3 Function Documentation

11.3.3.1 int ec_datagram_prealloc (ec_datagram_t * *datagram*, size_t *size*)

Allocates datagram data memory.

If the allocated memory is already larger than requested, nothing ist done.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
size New size in bytes

Definition at line 102 of file datagram.c.

11.3.3.2 int ec_datagram_nprd (ec_datagram_t * *datagram*, uint16_t *node_address*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT NPRD datagram.

Node-adressed physical read.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
node_address configured station address
offset physical memory address
data_size number of bytes to read

Definition at line 131 of file datagram.c.

11.3.3.3 int ec_datagram_npwr (ec_datagram_t * *datagram*, uint16_t *node_address*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT NPWR datagram.

Node-adressed physical write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
node_address configured station address
offset physical memory address
data_size number of bytes to write

Definition at line 159 of file datagram.c.

11.3.3.4 int ec_datagram_aprd (ec_datagram_t * *datagram*, uint16_t *ring_position*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT APRD datagram.

Autoincrement physical read.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
ring_position auto-increment position
offset physical memory address
data_size number of bytes to read

Definition at line 187 of file datagram.c.

11.3.3.5 int ec_datagram_apwr (ec_datagram_t * *datagram*, uint16_t *ring_position*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT APWR datagram.

Autoincrement physical write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
ring_position auto-increment position
offset physical memory address
data_size number of bytes to write

Definition at line 212 of file datagram.c.

11.3.3.6 int ec_datagram_brd (ec_datagram_t * *datagram*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT BRD datagram.

Broadcast read.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram
offset physical memory address
data_size number of bytes to read

Definition at line 237 of file datagram.c.

11.3.3.7 int ec_datagram_bwr (ec_datagram_t * *datagram*, uint16_t *offset*, size_t *data_size*)

Initializes an EtherCAT BWR datagram.

Broadcast write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram

offset physical memory address

data_size number of bytes to write

Definition at line 260 of file datagram.c.

11.3.3.8 int ec_datagram_lrw (ec_datagram_t * *datagram*, uint32_t *offset*, size_t *data_size*)

Initializes an EtherCAT LRW datagram.

Logical read write.

Returns:
0 in case of success, else < 0

Parameters:
datagram EtherCAT datagram

offset logical address

data_size number of bytes to read/write

Definition at line 283 of file datagram.c.

11.4 debug.c File Reference

11.4.1 Detailed Description

Ethernet interface for debugging purposes.

Definition in file **debug.c**.

Functions

- int **ec_dbgdev_open** (struct net_device *dev)
  Opens the virtual network device.
- int **ec_dbgdev_stop** (struct net_device *dev)
  Stops the virtual network device.
- net_device_stats * **ec_dbgdev_stats** (struct net_device *dev)
  Gets statistics about the virtual network device.
- int **ec_debug_init** (**ec_debug_t** *dbg)
  Debug constructor.
- void **ec_debug_clear** (**ec_debug_t** *dbg)
  Debug destructor.
- void **ec_debug_send** (**ec_debug_t** *dbg, const uint8_t *data, size_t size)
  Sends frame data to the interface.

11.4.2 Function Documentation

11.4.2.1 int ec_dbgdev_open (struct net_device *)

Opens the virtual network device.

Parameters:
dev debug net_device

Definition at line 155 of file debug.c.

11.4.2.2 int ec_dbgdev_stop (struct net_device *)

Stops the virtual network device.

Parameters:
dev debug net_device

Definition at line 169 of file debug.c.

11.4.2.3 struct net_device_stats * ec_dbgdev_stats (struct net_device *)

Gets statistics about the virtual network device.

Parameters:
dev debug net_device

Definition at line 183 of file debug.c.

11.4.2.4 int ec_debug_init (ec_debug_t * *dbg*)

Debug constructor.

Initializes the debug object, creates a net_device and registeres it.

Parameters:
dbg debug object

Definition at line 61 of file debug.c.

11.4.2.5 void ec_debug_clear (ec_debug_t * *dbg*)

Debug destructor.

Unregisteres the net_device and frees allocated memory.

Parameters:
dbg debug object

Definition at line 104 of file debug.c.

11.4.2.6 void ec_debug_send (ec_debug_t * *dbg*, const uint8_t * *data*, size_t *size*)

Sends frame data to the interface.

Parameters:
dbg debug object
data frame data
size size of the frame data

Definition at line 118 of file debug.c.

11.5 debug.h File Reference

11.5.1 Detailed Description

Network interface for debugging purposes.

Definition in file **debug.h**.

Data Structures

- struct **ec_debug_t**
  Debugging network interface.

Functions

- int **ec_debug_init** (**ec_debug_t** ∗)
  Debug constructor.
- void **ec_debug_clear** (**ec_debug_t** ∗)
  Debug destructor.
- void **ec_debug_send** (**ec_debug_t** ∗, const uint8_t ∗, size_t)
  Sends frame data to the interface.

11.5.2 Function Documentation

11.5.2.1 int ec_debug_init (ec_debug_t ∗ *dbg*)

Debug constructor.

Initializes the debug object, creates a net_device and registeres it.

Parameters:
dbg debug object

Definition at line 61 of file debug.c.

11.5.2.2 void ec_debug_clear (ec_debug_t ∗ *dbg*)

Debug destructor.

Unregisteres the net_device and frees allocated memory.

Parameters:
dbg debug object

Definition at line 104 of file debug.c.

11.6 device.c File Reference

11.6.1 Detailed Description

EtherCAT device methods.

Definition in file **device.c**.

Functions

- int **ec_device_init** (**ec_device_t** ∗device, **ec_master_t** ∗master, struct net_device ∗net_dev, **ec_isr_t** isr, struct module ∗module)
  Device constructor.
- void **ec_device_clear** (**ec_device_t** ∗device)
  EtherCAT device destuctor.
- int **ec_device_open** (**ec_device_t** ∗device)
  Opens the EtherCAT device.
- int **ec_device_close** (**ec_device_t** ∗device)
  Stops the EtherCAT device.
- uint8_t ∗ **ec_device_tx_data** (**ec_device_t** ∗device)
  Returns a pointer to the device's transmit memory.
- void **ec_device_send** (**ec_device_t** ∗device, size_t size)
  Sends the content of the transmit socket buffer.
- void **ec_device_call_isr** (**ec_device_t** ∗device)
  Calls the interrupt service routine of the assigned net_device.
- void **ecdev_receive** (**ec_device_t** ∗device, const void ∗data, size_t size)
  Accepts a received frame.
- void **ecdev_link_state** (**ec_device_t** ∗device, uint8_t state)
  Sets a new link state.

11.6.2 Function Documentation

11.6.2.1 int ec_device_init (ec_device_t ∗ *device*, ec_master_t ∗ *master*, struct net_device ∗ *net_dev*, ec_isr_t *isr*, struct module ∗ *module*)

Device constructor.

Returns:
0 in case of success, else < 0

Parameters:
device EtherCAT device

master master owning the device
net_dev net_device structure
isr pointer to device's ISR
module pointer to the owning module

Definition at line 57 of file device.c.

11.6.2.2 void ec_device_clear (ec_device_t * *device*)

EtherCAT device destuctor.

Parameters:
device EtherCAT device

Definition at line 107 of file device.c.

11.6.2.3 int ec_device_open (ec_device_t * *device*)

Opens the EtherCAT device.

Returns:
0 in case of success, else < 0

Parameters:
device EtherCAT device

Definition at line 121 of file device.c.

11.6.2.4 int ec_device_close (ec_device_t * *device*)

Stops the EtherCAT device.

Returns:
0 in case of success, else < 0

Parameters:
device EtherCAT device

Definition at line 152 of file device.c.

11.6.2.5 uint8_t* ec_device_tx_data (ec_device_t * *device*)

Returns a pointer to the device's transmit memory.

Returns:
pointer to the TX socket buffer

Parameters:
device EtherCAT device

Definition at line 176 of file device.c.

11.6.2.6 void ec_device_send (ec_device_t * *device*, size_t *size*)

Sends the content of the transmit socket buffer.

Cuts the socket buffer content to the (now known) size, and calls the start_xmit() function of the assigned net_device.

Parameters:

device EtherCAT device

size number of bytes to send

Definition at line 189 of file device.c.

11.6.2.7 void ec_device_call_isr (ec_device_t * *device*)

Calls the interrupt service routine of the assigned net_device.

The master itself works without using interrupts. Therefore the processing of received data and status changes of the network device has to be done by the master calling the ISR "manually".

Parameters:

device EtherCAT device

Definition at line 219 of file device.c.

11.7 device.h File Reference

11.7.1 Detailed Description

EtherCAT device structure.

Definition in file **device.h**.

Data Structures

- struct **ec_device**
  EtherCAT device.

Functions

- int **ec_device_init** (**ec_device_t** ∗, **ec_master_t** ∗, struct net_device ∗, **ec_isr_t**, struct module ∗)
  Device constructor.
- void **ec_device_clear** (**ec_device_t** ∗)
  EtherCAT device destuctor.
- int **ec_device_open** (**ec_device_t** ∗)
  Opens the EtherCAT device.
- int **ec_device_close** (**ec_device_t** ∗)
  Stops the EtherCAT device.
- void **ec_device_call_isr** (**ec_device_t** ∗)
  Calls the interrupt service routine of the assigned net_device.
- uint8_t ∗ **ec_device_tx_data** (**ec_device_t** ∗)
  Returns a pointer to the device's transmit memory.
- void **ec_device_send** (**ec_device_t** ∗, size_t)
  Sends the content of the transmit socket buffer.

11.7.2 Function Documentation

11.7.2.1 int ec_device_init (ec_device_t ∗ *device*, ec_master_t ∗ *master*, struct net_device ∗ *net_dev*, ec_isr_t *isr*, struct module ∗ *module*)

Device constructor.

Returns:
0 in case of success, else < 0

Parameters:
device EtherCAT device

master master owning the device
net_dev net_device structure
isr pointer to device's ISR
module pointer to the owning module

Definition at line 57 of file device.c.

11.7.2.2 int ec_device_open (ec_device_t * *device*)

Opens the EtherCAT device.

Returns:
0 in case of success, else < 0

Parameters:
device EtherCAT device

Definition at line 121 of file device.c.

11.7.2.3 int ec_device_close (ec_device_t * *device*)

Stops the EtherCAT device.

Returns:
0 in case of success, else < 0

Parameters:
device EtherCAT device

Definition at line 152 of file device.c.

11.7.2.4 void ec_device_call_isr (ec_device_t * *device*)

Calls the interrupt service routine of the assigned net_device.

The master itself works without using interrupts. Therefore the processing of received data and status changes of the network device has to be done by the master calling the ISR "manually".

Parameters:
device EtherCAT device

Definition at line 219 of file device.c.

11.7.2.5 uint8_t* ec_device_tx_data (ec_device_t * *device*)

Returns a pointer to the device's transmit memory.

Returns:
pointer to the TX socket buffer

Parameters:
device EtherCAT device

Definition at line 176 of file device.c.

11.7.2.6 void ec_device_send (ec_device_t * *device*, size_t *size*)

Sends the content of the transmit socket buffer.

Cuts the socket buffer content to the (now known) size, and calls the start_xmit() function of the assigned net_device.

Parameters:

device EtherCAT device

size number of bytes to send

Definition at line 189 of file device.c.

11.8 domain.c File Reference

11.8.1 Detailed Description

EtherCAT domain methods.

Definition in file **domain.c**.

Functions

- void **ec_domain_clear_field_regs** (**ec_domain_t** ∗domain)
  Clears the list of the registered data fields.
- ssize_t **ec_show_domain_attribute** (struct kobject ∗kobj,struct attribute ∗attr,char ∗buffer)
  Formats attribute data for SysFS reading.
- int **ec_domain_init** (**ec_domain_t** ∗domain, **ec_master_t** ∗master, unsigned int index)
  Domain constructor.
- void **ec_domain_clear** (struct kobject ∗kobj)
  Domain destructor.
- int **ec_domain_reg_field** (**ec_domain_t** ∗domain, **ec_slave_t** ∗slave, const **ec_sync_t** ∗sync, uint32_t field_offset, void ∗∗data_ptr)
  Registeres a data field in a domain.
- int **ec_domain_add_datagram** (**ec_domain_t** ∗domain, uint32_t offset, size_t data_size)
  Allocates a process data datagram and appends it to the list.
- int **ec_domain_alloc** (**ec_domain_t** ∗domain, uint32_t base_address)
  Creates a domain.
- void **ec_domain_response_count** (**ec_domain_t** ∗domain, unsigned int count)
  Sets the number of responding slaves and outputs it on demand.
- **ec_slave_t** ∗ **ecrt_domain_register_field** (**ec_domain_t** ∗domain, const char ∗address, const char ∗vendor_name, const char ∗product_name, void ∗∗data_ptr, const char ∗field_name, unsigned int field_index, unsigned int field_count)
  Registers a data field in a domain.
- int **ecrt_domain_register_field_list** (**ec_domain_t** ∗domain, const **ec_field_init_t** ∗fields)
  Registeres a bunch of data fields.
- void **ecrt_domain_queue** (**ec_domain_t** ∗domain)
  Places all process data datagrams in the masters datagram queue.
- void **ecrt_domain_process** (**ec_domain_t** ∗domain)
  Processes received process data.
- int **ecrt_domain_state** (**ec_domain_t** ∗domain)
  Returns the state of a domain.

11.8.2 Function Documentation

11.8.2.1 void ec_domain_clear_field_regs (ec_domain_t *)

Clears the list of the registered data fields.

Parameters:
domain EtherCAT domain

Definition at line 179 of file domain.c.

11.8.2.2 ssize_t ec_show_domain_attribute (struct kobject * *kobj*, struct attribute * *attr*, char * *buffer*)

Formats attribute data for SysFS reading.

Returns:
number of bytes to read

Parameters:
kobj kobject

attr attribute

buffer memory to store data in

Definition at line 341 of file domain.c.

11.8.2.3 int ec_domain_init (ec_domain_t * *domain*, ec_master_t * *master*, unsigned int *index*)

Domain constructor.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain

master owning master

index domain index

Definition at line 83 of file domain.c.

11.8.2.4 void ec_domain_clear (struct kobject * *kobj*)

Domain destructor.

Parameters:
kobj kobject of the domain

Definition at line 116 of file domain.c.

11.8.2.5 int ec_domain_reg_field (ec_domain_t * *domain*, ec_slave_t * *slave*, const ec_sync_t * *sync*, uint32_t *field_offset*, void ** *data_ptr*)

Registeres a data field in a domain.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain

slave slave

sync sync manager

field_offset data field offset

data_ptr pointer to the process data pointer

Definition at line 142 of file domain.c.

11.8.2.6 int ec_domain_add_datagram (ec_domain_t * *domain*, uint32_t *offset*, size_t *data_size*)

Allocates a process data datagram and appends it to the list.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain

offset logical offset

data_size size of the datagram data

Definition at line 196 of file domain.c.

11.8.2.7 int ec_domain_alloc (ec_domain_t * *domain*, uint32_t *base_address*)

Creates a domain.

Reserves domain memory, calculates the logical addresses of the corresponding FMMUs and sets the process data pointer of the registered data fields.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain

base_address logical base address

Definition at line 229 of file domain.c.

11.8.2.8 void ec_domain_response_count (ec_domain_t * *domain*, unsigned int *count*)

Sets the number of responding slaves and outputs it on demand.

This number isn't really the number of responding slaves, but the sum of the working counters of all domain datagrams. Some slaves increase the working counter by 2, some by 1.

Parameters:

domain EtherCAT domain

count new WC sum

Definition at line 323 of file domain.c.

11.9 domain.h File Reference

11.9.1 Detailed Description

EtherCAT domain structure.

Definition in file **domain.h**.

Data Structures

- struct **ec_field_reg_t**

  Data field registration type.

- struct **ec_domain**

  EtherCAT domain.

Functions

- int **ec_domain_init** (**ec_domain_t** ∗, **ec_master_t** ∗, unsigned int)

  Domain constructor.

- void **ec_domain_clear** (struct kobject ∗)

  Domain destructor.

- int **ec_domain_alloc** (**ec_domain_t** ∗, uint32_t)

  Creates a domain.

11.9.2 Function Documentation

11.9.2.1 int ec_domain_init (ec_domain_t ∗ *domain*, ec_master_t ∗ *master*, unsigned int *index*)

Domain constructor.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain

master owning master

index domain index

Definition at line 83 of file domain.c.

11.9.2.2 int ec_domain_alloc (ec_domain_t * *domain*, uint32_t *base_address*)

Creates a domain.

Reserves domain memory, calculates the logical addresses of the corresponding FMMUs and sets the process data pointer of the registered data fields.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain

base_address logical base address

Definition at line 229 of file domain.c.

11.10 ecdev.h File Reference

11.10.1 Detailed Description

EtherCAT interface for EtherCAT device drivers.

Definition in file **ecdev.h**.

Typedefs

- typedef **ec_device ec_device_t**
- typedef irqreturn_t(∗ **ec_isr_t**)(int, void ∗, struct pt_regs ∗)

  Interrupt-Service-Routine Type.

Functions

- **ec_device_t** ∗ **ecdev_register** (unsigned int master_index, struct net_device ∗net_dev, **ec_isr_t** isr, struct module ∗module)

  Connects an EtherCAT device to a certain master.

- void **ecdev_unregister** (unsigned int master_index, **ec_device_t** ∗device)

  Disconnect an EtherCAT device from the master.

- int **ecdev_start** (unsigned int master_index)

  Starts the master associated with the device.

- void **ecdev_stop** (unsigned int master_index)

  Stops the master associated with the device.

- void **ecdev_receive** (**ec_device_t** ∗device, const void ∗data, size_t size)

  Accepts a received frame.

- void **ecdev_link_state** (**ec_device_t** ∗device, uint8_t newstate)

  Sets a new link state.

11.10.2 Typedef Documentation

11.10.2.1 typedef struct ec_device ec_device_t

See also:
ec_device(p. 37)

Definition at line 58 of file ecdev.h.

11.11 ecrt.h File Reference

11.11.1 Detailed Description

EtherCAT realtime interface.

Definition in file **ecrt.h**.

Data Structures

- struct **ec_field_init_t**
  Initialization type for field registrations.

Defines

- #define **EC_READ_BIT**(DATA, POS) ((∗((uint8_t ∗) (DATA)) >> (POS)) & 0x01)
  Read a certain bit of an EtherCAT data byte.
- #define **EC_WRITE_BIT**(DATA, POS, VAL)
  Write a certain bit of an EtherCAT data byte.
- #define **EC_READ_U8**(DATA) ((uint8_t) ∗((uint8_t ∗) (DATA)))
  Read an 8-bit unsigned value from EtherCAT data.
- #define **EC_READ_S8**(DATA) ((int8_t) ∗((uint8_t ∗) (DATA)))
  Read an 8-bit signed value from EtherCAT data.
- #define **EC_READ_U16**(DATA) ((uint16_t) le16_to_cpup((void ∗) (DATA)))
  Read a 16-bit unsigned value from EtherCAT data.
- #define **EC_READ_S16**(DATA) ((int16_t) le16_to_cpup((void ∗) (DATA)))
  Read a 16-bit signed value from EtherCAT data.
- #define **EC_READ_U32**(DATA) ((uint32_t) le32_to_cpup((void ∗) (DATA)))
  Read a 32-bit unsigned value from EtherCAT data.
- #define **EC_READ_S32**(DATA) ((int32_t) le32_to_cpup((void ∗) (DATA)))
  Read a 32-bit signed value from EtherCAT data.
- #define **EC_WRITE_U8**(DATA, VAL)
  Write an 8-bit unsigned value to EtherCAT data.
- #define **EC_WRITE_S8**(DATA, VAL) EC_WRITE_U8(DATA, VAL)
  Write an 8-bit signed value to EtherCAT data.
- #define **EC_WRITE_U16**(DATA, VAL)
  Write a 16-bit unsigned value to EtherCAT data.

- #define **EC_WRITE_S16**(DATA, VAL) EC_WRITE_U16(DATA, VAL)
  Write a 16-bit signed value to EtherCAT data.
- #define **EC_WRITE_U32**(DATA, VAL)
  Write a 32-bit unsigned value to EtherCAT data.
- #define **EC_WRITE_S32**(DATA, VAL) EC_WRITE_U32(DATA, VAL)
  Write a 32-bit signed value to EtherCAT data.

Typedefs

- typedef **ec_master ec_master_t**
- typedef **ec_domain ec_domain_t**
- typedef **ec_slave ec_slave_t**

Functions

- **ec_master_t** ∗ **ecrt_request_master** (unsigned int master_index)
  Reserves an EtherCAT master for realtime operation.
- void **ecrt_release_master** (**ec_master_t** ∗master)
  Releases a reserved EtherCAT master.
- void **ecrt_master_callbacks** (**ec_master_t** ∗master, int(∗request_cb)(void ∗), void(∗release_cb)(void ∗), void ∗cb_data)
  Sets the locking callbacks.
- **ec_domain_t** ∗ **ecrt_master_create_domain** (**ec_master_t** ∗master)
  Creates a domain.
- int **ecrt_master_activate** (**ec_master_t** ∗master)
  Configures all slaves and leads them to the OP state.
- void **ecrt_master_deactivate** (**ec_master_t** ∗master)
  Resets all slaves to INIT state.
- int **ecrt_master_fetch_sdo_lists** (**ec_master_t** ∗master)
  Fetches the SDO dictionaries of all slaves.
- void **ecrt_master_sync_io** (**ec_master_t** ∗master)
  Sends queued datagrams and waits for their reception.
- void **ecrt_master_async_send** (**ec_master_t** ∗master)
  Asynchronous sending of datagrams.
- void **ecrt_master_async_receive** (**ec_master_t** ∗master)
  Asynchronous receiving of datagrams.

- void **ecrt_master_prepare_async_io** (**ec_master_t** *master)

  Prepares synchronous IO.

- void **ecrt_master_run** (**ec_master_t** *master)

  Does all cyclic master work.

- int **ecrt_master_start_eoe** (**ec_master_t** *master)

  Starts Ethernet-over-EtherCAT processing for all EoE-capable slaves.

- void **ecrt_master_debug** (**ec_master_t** *master, int level)

  Sets the debug level of the master.

- void **ecrt_master_print** (const **ec_master_t** *master, unsigned int verbosity)

  Outputs all master information.

- **ec_slave_t** * **ecrt_master_get_slave** (const **ec_master_t** *, const char *)

  Translates an ASCII coded bus-address to a slave pointer.

- **ec_slave_t** * **ecrt_domain_register_field** (**ec_domain_t** *domain, const char *address, const char *vendor_name, const char *product_name, void **data_ptr, const char *field_name, unsigned int field_index, unsigned int field_count)

  Registers a data field in a domain.

- int **ecrt_domain_register_field_list** (**ec_domain_t** *domain, const **ec_field_init_t** *fields)

  Registeres a bunch of data fields.

- void **ecrt_domain_queue** (**ec_domain_t** *domain)

  Places all process data datagrams in the masters datagram queue.

- void **ecrt_domain_process** (**ec_domain_t** *domain)

  Processes received process data.

- int **ecrt_domain_state** (**ec_domain_t** *domain)

  Returns the state of a domain.

- int **ecrt_slave_sdo_read_exp8** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t *value)

  Reads an 8-bit SDO in expedited mode.

- int **ecrt_slave_sdo_read_exp16** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t *value)

  Reads a 16-bit SDO in expedited mode.

- int **ecrt_slave_sdo_read_exp32** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t *value)

  Reads a 32-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp8** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t value)

  Writes an 8-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp16** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t value)

  Writes a 16-bit SDO in expedited mode.

- int **ecrt_slave_sdo_write_exp32** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t value)

  Writes a 32-bit SDO in expedited mode.

- int **ecrt_slave_sdo_read** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t ∗data, size_t ∗size)

  Reads a CANopen SDO in normal mode.

- int **ecrt_slave_write_alias** (**ec_slave_t** ∗slave, uint16_t alias)

  Writes the "configured station alias" to the slave's EEPROM.

- int **ecrt_slave_field_size** (**ec_slave_t** ∗slave, const char ∗field_name, unsigned int field_index, size_t size)

11.11.2 Define Documentation

11.11.2.1 #define EC_READ_BIT(DATA, POS) ((∗((uint8_t ∗) (DATA)) >> (POS)) & 0x01)

Read a certain bit of an EtherCAT data byte.

Parameters:

DATA EtherCAT data pointer

POS bit position

Definition at line 180 of file ecrt.h.

11.11.2.2 #define EC_WRITE_BIT(DATA, POS, VAL)

Value:

```
do { \
        if (VAL) *((uint8_t *) (DATA)) |=  (1 << (POS)); \
        else     *((uint8_t *) (DATA)) &= ~(1 << (POS)); \
    } while (0)
```

Write a certain bit of an EtherCAT data byte.

Parameters:

DATA EtherCAT data pointer

POS bit position

VAL new bit value

Definition at line 189 of file ecrt.h.

11.11.2.3 #define EC_READ_U8(DATA) ((uint8_t) *((uint8_t *) (DATA)))

Read an 8-bit unsigned value from EtherCAT data.

Returns:
EtherCAT data value

Definition at line 204 of file ecrt.h.

11.11.2.4 #define EC_READ_S8(DATA) ((int8_t) *((uint8_t *) (DATA)))

Read an 8-bit signed value from EtherCAT data.

Parameters:
DATA EtherCAT data pointer

Returns:
EtherCAT data value

Definition at line 213 of file ecrt.h.

11.11.2.5 #define EC_READ_U16(DATA) ((uint16_t) le16_to_cpup((void *) (DATA)))

Read a 16-bit unsigned value from EtherCAT data.

Parameters:
DATA EtherCAT data pointer

Returns:
EtherCAT data value

Definition at line 222 of file ecrt.h.

11.11.2.6 #define EC_READ_S16(DATA) ((int16_t) le16_to_cpup((void *) (DATA)))

Read a 16-bit signed value from EtherCAT data.

Parameters:
DATA EtherCAT data pointer

Returns:
EtherCAT data value

Definition at line 231 of file ecrt.h.

11.11.2.7 #define EC_READ_U32(DATA) ((uint32_t) le32_to_cpup((void *) (DATA)))

Read a 32-bit unsigned value from EtherCAT data.

Parameters:
DATA EtherCAT data pointer

Returns:
EtherCAT data value

Definition at line 240 of file ecrt.h.

11.11.2.8 #define EC_READ_S32(DATA) ((int32_t) le32_to_cpup((void *) (DATA)))

Read a 32-bit signed value from EtherCAT data.

Parameters:
DATA EtherCAT data pointer

Returns:
EtherCAT data value

Definition at line 249 of file ecrt.h.

11.11.2.9 #define EC_WRITE_U8(DATA, VAL)

Value:

```
do { \
        *((uint8_t *)(DATA)) = ((uint8_t) (VAL)); \
    } while (0)
```

Write an 8-bit unsigned value to EtherCAT data.

Parameters:
DATA EtherCAT data pointer
VAL new value

Definition at line 263 of file ecrt.h.

11.11.2.10 #define EC_WRITE_S8(DATA, VAL) EC_WRITE_U8(DATA, VAL)

Write an 8-bit signed value to EtherCAT data.

Parameters:
DATA EtherCAT data pointer
VAL new value

Definition at line 274 of file ecrt.h.

11.11.2.11 #define EC_WRITE_U16(DATA, VAL)

Value:

```
do { \
        *((uint16_t *) (DATA)) = (uint16_t) (VAL); \
        cpu_to_le16s(DATA); \
    } while (0)
```

Write a 16-bit unsigned value to EtherCAT data.

Parameters:

DATA EtherCAT data pointer

VAL new value

Definition at line 282 of file ecrt.h.

11.11.2.12 #define EC_WRITE_S16(DATA, VAL) EC_WRITE_U16(DATA, VAL)

Write a 16-bit signed value to EtherCAT data.

Parameters:

DATA EtherCAT data pointer

VAL new value

Definition at line 294 of file ecrt.h.

11.11.2.13 #define EC_WRITE_U32(DATA, VAL)

Value:

```
do { \
        *((uint32_t *) (DATA)) = (uint32_t) (VAL); \
        cpu_to_le16s(DATA); \
    } while (0)
```

Write a 32-bit unsigned value to EtherCAT data.

Parameters:

DATA EtherCAT data pointer

VAL new value

Definition at line 302 of file ecrt.h.

11.11.2.14 #define EC_WRITE_S32(DATA, VAL) EC_WRITE_U32(DATA, VAL)

Write a 32-bit signed value to EtherCAT data.

Parameters:

DATA EtherCAT data pointer

VAL new value

Definition at line 314 of file ecrt.h.

11.11.3 Typedef Documentation

11.11.3.1 typedef struct ec_master ec_master_t

See also:
ec_master(p. 52)

Definition at line 63 of file ecrt.h.

11.11.3.2 typedef struct ec_domain ec_domain_t

See also:
ec_domain(p. 38)

Definition at line 66 of file ecrt.h.

11.11.3.3 typedef struct ec_slave ec_slave_t

See also:
ec_slave(p. 56)

Definition at line 69 of file ecrt.h.

11.12 ethernet.c File Reference

11.12.1 Detailed Description

Ethernet-over-EtherCAT (EoE).

Definition in file **ethernet.c**.

Defines

- #define **EOE_DEBUG_LEVEL** 0

  Defines the debug level of EoE processing.

Functions

- void **ec_eoe_flush** (**ec_eoe_t** ∗eoe)

  Empties the transmit queue.

- void **ec_eoe_state_rx_start** (**ec_eoe_t** ∗eoe)

  State: RX_START.

- void **ec_eoe_state_rx_check** (**ec_eoe_t** ∗eoe)

  State: RX_CHECK.

- void **ec_eoe_state_rx_fetch** (**ec_eoe_t** ∗eoe)

  State: RX_FETCH.

- void **ec_eoe_state_tx_start** (**ec_eoe_t** ∗eoe)

  State: TX START.

- void **ec_eoe_state_tx_sent** (**ec_eoe_t** ∗eoe)

  State: TX SENT.

- int **ec_eoedev_open** (struct net_device ∗dev)

  Opens the virtual network device.

- int **ec_eoedev_stop** (struct net_device ∗dev)

  Stops the virtual network device.

- int **ec_eoedev_tx** (struct sk_buff ∗skb,struct net_device ∗dev)

  Transmits data via the virtual network device.

- net_device_stats ∗ **ec_eoedev_stats** (struct net_device ∗dev)

  Gets statistics about the virtual network device.

- int **ec_eoe_init** (**ec_eoe_t** ∗eoe)

  EoE constructor.

- void **ec_eoe_clear** (**ec_eoe_t** ∗eoe)

  EoE destructor.

- int **ec_eoe_send** (**ec_eoe_t** ∗eoe)

  Sends a frame or the next fragment.

- void **ec_eoe_run** (**ec_eoe_t** ∗eoe)

  Runs the EoE state machine.

- unsigned int **ec_eoe_active** (const **ec_eoe_t** ∗eoe)

  Returns the state of the device.

- void **ec_eoe_print** (const **ec_eoe_t** ∗eoe)

  Prints EoE handler information.

11.12.2 Define Documentation

11.12.2.1 #define EOE_DEBUG_LEVEL 0

Defines the debug level of EoE processing.

0 = No debug messages. 1 = Output actions. 2 = Output actions and frame data.

Definition at line 59 of file ethernet.c.

11.12.3 Function Documentation

11.12.3.1 void ec_eoe_flush (ec_eoe_t ∗)

Empties the transmit queue.

Parameters:
eoe EoE handler

Definition at line 182 of file ethernet.c.

11.12.3.2 void ec_eoe_state_rx_start (ec_eoe_t ∗ *eoe*)

State: RX_START.

Starts a new receiving sequence by queueing a datagram that checks the slave's mailbox for a new EoE datagram.

Parameters:
eoe EoE handler

Definition at line 321 of file ethernet.c.

11.12.3.3 void ec_eoe_state_rx_check (ec_eoe_t * *eoe*)

State: RX_CHECK.

Processes the checking datagram sent in RX_START and issues a receive datagram, if new data is available.

Parameters:
eoe EoE handler

Definition at line 339 of file ethernet.c.

11.12.3.4 void ec_eoe_state_rx_fetch (ec_eoe_t * *eoe*)

State: RX_FETCH.

Checks if the requested data of RX_CHECK was received and processes the EoE datagram.

Parameters:
eoe EoE handler

Definition at line 365 of file ethernet.c.

11.12.3.5 void ec_eoe_state_tx_start (ec_eoe_t * *eoe*)

State: TX START.

Starts a new transmit sequence. If no data is available, a new receive sequence is started instead.

Parameters:
eoe EoE handler

Definition at line 508 of file ethernet.c.

11.12.3.6 void ec_eoe_state_tx_sent (ec_eoe_t * *eoe*)

State: TX SENT.

Checks is the previous transmit datagram succeded and sends the next fragment, if necessary.

Parameters:
eoe EoE handler

Definition at line 571 of file ethernet.c.

11.12.3.7 int ec_eoedev_open (struct net_device *)

Opens the virtual network device.

Parameters:
dev EoE net_device

Definition at line 613 of file ethernet.c.

11.12.3.8 int ec_eoedev_stop (struct net_device *)

Stops the virtual network device.

Parameters:
dev EoE net_device

Definition at line 636 of file ethernet.c.

11.12.3.9 int ec_eoedev_tx (struct sk_buff *, struct net_device *)

Transmits data via the virtual network device.

Parameters:
skb transmit socket buffer

dev EoE net_device

Definition at line 659 of file ethernet.c.

11.12.3.10 struct net_device_stats * ec_eoedev_stats (struct net_device *)

Gets statistics about the virtual network device.

Parameters:
dev EoE net_device

Definition at line 709 of file ethernet.c.

11.12.3.11 int ec_eoe_init (ec_eoe_t * *eoe*)

EoE constructor.

Initializes the EoE handler, creates a net_device and registeres it.

Parameters:
eoe EoE handler

Definition at line 85 of file ethernet.c.

11.12.3.12 void ec_eoe_clear (ec_eoe_t * *eoe*)

EoE destructor.

Unregisteres the net_device and frees allocated memory.

Parameters:
eoe EoE handler

Definition at line 156 of file ethernet.c.

11.12.3.13 int ec_eoe_send (ec_eoe_t * *eoe*)

Sends a frame or the next fragment.

Parameters:
eoe EoE handler

Definition at line 204 of file ethernet.c.

11.12.3.14 void ec_eoe_run (ec_eoe_t * *eoe*)

Runs the EoE state machine.

Parameters:
eoe EoE handler

Definition at line 275 of file ethernet.c.

11.12.3.15 unsigned int ec_eoe_active (const ec_eoe_t * *eoe*)

Returns the state of the device.

Returns:
1 if the device is "up", 0 if it is "down"

Parameters:
eoe EoE handler

Definition at line 290 of file ethernet.c.

11.12.3.16 void ec_eoe_print (const ec_eoe_t * *eoe*)

Prints EoE handler information.

Parameters:
eoe EoE handler

Definition at line 301 of file ethernet.c.

11.13 ethernet.h File Reference

11.13.1 Detailed Description

Ethernet-over-EtherCAT (EoE).

Definition in file **ethernet.h**.

Data Structures

- struct **ec_eoe_frame_t**

  Queued frame structure.

- struct **ec_eoe**

  Ethernet-over-EtherCAT (EoE) handler.

Typedefs

- typedef **ec_eoe ec_eoe_t**

Functions

- int **ec_eoe_init** (**ec_eoe_t** ∗)

  EoE constructor.

- void **ec_eoe_clear** (**ec_eoe_t** ∗)

  EoE destructor.

- void **ec_eoe_run** (**ec_eoe_t** ∗)

  Runs the EoE state machine.

- unsigned int **ec_eoe_active** (const **ec_eoe_t** ∗)

  Returns the state of the device.

- void **ec_eoe_print** (const **ec_eoe_t** ∗)

  Prints EoE handler information.

11.13.2 Typedef Documentation

11.13.2.1 typedef struct ec_eoe ec_eoe_t

See also:
ec_eoe(p. 43)

Definition at line 64 of file ethernet.h.

11.13.3 Function Documentation

11.13.3.1 int ec_eoe_init (ec_eoe_t * *eoe*)

EoE constructor.

Initializes the EoE handler, creates a net_device and registeres it.

Parameters:
eoe EoE handler

Definition at line 85 of file ethernet.c.

11.13.3.2 void ec_eoe_clear (ec_eoe_t * *eoe*)

EoE destructor.

Unregisteres the net_device and frees allocated memory.

Parameters:
eoe EoE handler

Definition at line 156 of file ethernet.c.

11.13.3.3 unsigned int ec_eoe_active (const ec_eoe_t * *eoe*)

Returns the state of the device.

Returns:
1 if the device is "up", 0 if it is "down"

Parameters:
eoe EoE handler

Definition at line 290 of file ethernet.c.

11.14 fsm.c File Reference

11.14.1 Detailed Description

EtherCAT finite state machines.

Definition in file **fsm.c**.

Functions

- void **ec_fsm_master_start** (**ec_fsm_t** ∗fsm)
  Master state: START.
- void **ec_fsm_master_broadcast** (**ec_fsm_t** ∗fsm)
  Master state: BROADCAST.
- void **ec_fsm_master_read_states** (**ec_fsm_t** ∗fsm)
  Master state: STATES.
- void **ec_fsm_master_validate_vendor** (**ec_fsm_t** ∗fsm)
  Master state: VALIDATE_VENDOR.
- void **ec_fsm_master_validate_product** (**ec_fsm_t** ∗fsm)
  Master state: VALIDATE_PRODUCT.
- void **ec_fsm_master_rewrite_addresses** (**ec_fsm_t** ∗fsm)
  Master state: ADDRESS.
- void **ec_fsm_master_configure_slave** (**ec_fsm_t** ∗fsm)
  Master state: CONF.
- void **ec_fsm_master_scan_slaves** (**ec_fsm_t** ∗fsm)
  Master state: SCAN.
- void **ec_fsm_master_write_eeprom** (**ec_fsm_t** ∗fsm)
  Master state: EEPROM.
- void **ec_fsm_slavescan_start** (**ec_fsm_t** ∗fsm)
  Slave state: START_READING.
- void **ec_fsm_slavescan_address** (**ec_fsm_t** ∗fsm)
  Slave state: ADDRESS.
- void **ec_fsm_slavescan_state** (**ec_fsm_t** ∗fsm)
  Slave state: STATE.
- void **ec_fsm_slavescan_base** (**ec_fsm_t** ∗fsm)
  Slave state: BASE.
- void **ec_fsm_slavescan_datalink** (**ec_fsm_t** ∗fsm)

Slave state: DATALINK.

- void **ec_fsm_slavescan_eeprom_size** (**ec_fsm_t** *fsm)
  Slave state: EEPROM_SIZE.
- void **ec_fsm_slavescan_eeprom_data** (**ec_fsm_t** *fsm)
  Slave state: EEPROM_DATA.
- void **ec_fsm_slavescan_end** (**ec_fsm_t** *fsm)
  Slave state: END.
- void **ec_fsm_slaveconf_init** (**ec_fsm_t** *fsm)
  Slave state: INIT.
- void **ec_fsm_slaveconf_sync** (**ec_fsm_t** *fsm)
  Slave state: SYNC.
- void **ec_fsm_slaveconf_preop** (**ec_fsm_t** *fsm)
  Slave state: PREOP.
- void **ec_fsm_slaveconf_fmmu** (**ec_fsm_t** *fsm)
  Slave state: FMMU.
- void **ec_fsm_slaveconf_saveop** (**ec_fsm_t** *fsm)
  Slave state: SAVEOP.
- void **ec_fsm_slaveconf_op** (**ec_fsm_t** *fsm)
  Slave state: OP.
- void **ec_fsm_slaveconf_end** (**ec_fsm_t** *fsm)
  Slave state: END.
- void **ec_fsm_sii_start_reading** (**ec_fsm_t** *fsm)
  SII state: START_READING.
- void **ec_fsm_sii_read_check** (**ec_fsm_t** *fsm)
  SII state: READ_CHECK.
- void **ec_fsm_sii_read_fetch** (**ec_fsm_t** *fsm)
  SII state: READ_FETCH.
- void **ec_fsm_sii_start_writing** (**ec_fsm_t** *fsm)
  SII state: START_WRITING.
- void **ec_fsm_sii_write_check** (**ec_fsm_t** *fsm)
  SII state: WRITE_CHECK.
- void **ec_fsm_sii_write_check2** (**ec_fsm_t** *fsm)
  SII state: WRITE_CHECK2.

- void **ec_fsm_sii_end** (**ec_fsm_t** ∗fsm)
  SII state: END.
- void **ec_fsm_sii_error** (**ec_fsm_t** ∗fsm)
  SII state: ERROR.
- void **ec_fsm_change_start** (**ec_fsm_t** ∗fsm)
  Change state: START.
- void **ec_fsm_change_check** (**ec_fsm_t** ∗fsm)
  Change state: CHECK.
- void **ec_fsm_change_status** (**ec_fsm_t** ∗fsm)
  Change state: STATUS.
- void **ec_fsm_change_code** (**ec_fsm_t** ∗fsm)
  Change state: CODE.
- void **ec_fsm_change_ack** (**ec_fsm_t** ∗fsm)
  Change state: ACK.
- void **ec_fsm_change_check_ack** (**ec_fsm_t** ∗fsm)
  Change state: CHECK ACK.
- void **ec_fsm_change_end** (**ec_fsm_t** ∗fsm)
  Change state: END.
- void **ec_fsm_change_error** (**ec_fsm_t** ∗fsm)
  Change state: ERROR.
- int **ec_fsm_init** (**ec_fsm_t** ∗fsm, **ec_master_t** ∗master)
  Constructor.
- void **ec_fsm_clear** (**ec_fsm_t** ∗fsm)
  Destructor.
- void **ec_fsm_reset** (**ec_fsm_t** ∗fsm)
  Resets the state machine.
- void **ec_fsm_execute** (**ec_fsm_t** ∗fsm)
  Executes the current state of the state machine.
- void **ec_fsm_master_action_process_states** (**ec_fsm_t** ∗fsm)
  Master action: PROC_STATES.
- void **ec_fsm_master_action_next_slave_state** (**ec_fsm_t** ∗fsm)
  Master action: Get state of next slave.
- void **ec_fsm_master_action_addresses** (**ec_fsm_t** ∗fsm)
  Master action: ADDRESS.

Variables

- const **ec_code_msg_t al_status_messages** [ ]

  Application layer status messages.

11.14.2 Function Documentation

11.14.2.1 void ec_fsm_master_start (ec_fsm_t * *fsm*)

Master state: START.

Starts with getting slave count and slave states.

Definition at line 161 of file fsm.c.

11.14.2.2 void ec_fsm_master_broadcast (ec_fsm_t * *fsm*)

Master state: BROADCAST.

Processes the broadcast read slave count and slaves states.

Parameters:
fsm finite state machine

Definition at line 175 of file fsm.c.

11.14.2.3 void ec_fsm_master_read_states (ec_fsm_t * *fsm*)

Master state: STATES.

Fetches the AL- and online state of a slave.

Parameters:
fsm finite state machine

Definition at line 398 of file fsm.c.

11.14.2.4 void ec_fsm_master_validate_vendor (ec_fsm_t * *fsm*)

Master state: VALIDATE_VENDOR.

Validates the vendor ID of a slave.

Parameters:
fsm finite state machine

Definition at line 449 of file fsm.c.

11.14.2.5 void ec_fsm_master_validate_product (ec_fsm_t * *fsm*)

Master state: VALIDATE_PRODUCT.

Validates the product ID of a slave.

Parameters:
fsm finite state machine

Definition at line 519 of file fsm.c.

11.14.2.6 void ec_fsm_master_rewrite_addresses (ec_fsm_t * *fsm*)

Master state: ADDRESS.

Checks, if the new station address has been written to the slave.

Parameters:
fsm finite state machine

Definition at line 569 of file fsm.c.

11.14.2.7 void ec_fsm_master_configure_slave (ec_fsm_t * *fsm*)

Master state: CONF.

Starts configuring a slave.

Parameters:
fsm finite state machine

Definition at line 696 of file fsm.c.

11.14.2.8 void ec_fsm_master_scan_slaves (ec_fsm_t * *fsm*)

Master state: SCAN.

Executes the sub-statemachine for the scanning of a slave.

Parameters:
fsm finite state machine

Definition at line 600 of file fsm.c.

11.14.2.9 void ec_fsm_master_write_eeprom (ec_fsm_t *)

Master state: EEPROM.

Parameters:
fsm finite state machine

Definition at line 712 of file fsm.c.

11.14.2.10 void ec_fsm_slavescan_start (ec_fsm_t * *fsm*)

Slave state: START_READING.

First state of the slave state machine. Writes the station address to the slave, according to its ring position.

Parameters:
fsm finite state machine

Definition at line 760 of file fsm.c.

11.14.2.11 void ec_fsm_slavescan_address (ec_fsm_t *)

Slave state: ADDRESS.

Parameters:
fsm finite state machine

Definition at line 777 of file fsm.c.

11.14.2.12 void ec_fsm_slavescan_state (ec_fsm_t *)

Slave state: STATE.

Parameters:
fsm finite state machine

Definition at line 801 of file fsm.c.

11.14.2.13 void ec_fsm_slavescan_base (ec_fsm_t *)

Slave state: BASE.

Parameters:
fsm finite state machine

Definition at line 833 of file fsm.c.

11.14.2.14 void ec_fsm_slavescan_datalink (ec_fsm_t *)

Slave state: DATALINK.

Parameters:
fsm finite state machine

Definition at line 867 of file fsm.c.

11.14.2.15 void ec_fsm_slavescan_eeprom_size (ec_fsm_t *)

Slave state: EEPROM_SIZE.

Parameters:
fsm finite state machine

Definition at line 904 of file fsm.c.

11.14.2.16 void ec_fsm_slavescan_eeprom_data (ec_fsm_t *)

Slave state: EEPROM_DATA.

Parameters:
fsm finite state machine

Definition at line 964 of file fsm.c.

11.14.2.17 void ec_fsm_slavescan_end (ec_fsm_t * *fsm*)

Slave state: END.

End state of the slave state machine.

Parameters:
fsm finite state machine

Definition at line 1076 of file fsm.c.

11.14.2.18 void ec_fsm_slaveconf_init (ec_fsm_t *)

Slave state: INIT.

Parameters:
fsm finite state machine

Definition at line 1088 of file fsm.c.

11.14.2.19 void ec_fsm_slaveconf_sync (ec_fsm_t *)

Slave state: SYNC.

Parameters:
fsm finite state machine

Definition at line 1190 of file fsm.c.

11.14.2.20 void ec_fsm_slaveconf_preop (ec_fsm_t *)

Slave state: PREOP.

Parameters:
fsm finite state machine

Definition at line 1215 of file fsm.c.

11.14.2.21 void ec_fsm_slaveconf_fmmu (ec_fsm_t *)

Slave state: FMMU.

Parameters:
fsm finite state machine

Definition at line 1271 of file fsm.c.

11.14.2.22 void ec_fsm_slaveconf_saveop (ec_fsm_t *)

Slave state: SAVEOP.

Parameters:
fsm finite state machine

Definition at line 1296 of file fsm.c.

11.14.2.23 void ec_fsm_slaveconf_op (ec_fsm_t *)

Slave state: OP.

Parameters:
fsm finite state machine

Definition at line 1327 of file fsm.c.

11.14.2.24 void ec_fsm_slaveconf_end (ec_fsm_t * *fsm*)

Slave state: END.

End state of the slave state machine.

Parameters:
fsm finite state machine

Definition at line 1350 of file fsm.c.

11.14.2.25 void ec_fsm_sii_start_reading (ec_fsm_t * *fsm*)

SII state: START_READING.

Starts reading the slave information interface.

Parameters:
fsm finite state machine

Definition at line 1363 of file fsm.c.

11.14.2.26 void ec_fsm_sii_read_check (ec_fsm_t * *fsm*)

SII state: READ_CHECK.

Checks, if the SII-read-datagram has been sent and issues a fetch datagram.

Parameters:
fsm finite state machine

Definition at line 1389 of file fsm.c.

11.14.2.27 void ec_fsm_sii_read_fetch (ec_fsm_t * *fsm*)

SII state: READ_FETCH.

Fetches the result of an SII-read datagram.

Parameters:
fsm finite state machine

Definition at line 1420 of file fsm.c.

11.14.2.28 void ec_fsm_sii_start_writing (ec_fsm_t * *fsm*)

SII state: START_WRITING.

Starts reading the slave information interface.

Parameters:
fsm finite state machine

Definition at line 1476 of file fsm.c.

11.14.2.29 void ec_fsm_sii_write_check (ec_fsm_t *)

SII state: WRITE_CHECK.

Parameters:
fsm finite state machine

Definition at line 1496 of file fsm.c.

11.14.2.30 void ec_fsm_sii_write_check2 (ec_fsm_t *)

SII state: WRITE_CHECK2.

Parameters:
fsm finite state machine

Definition at line 1520 of file fsm.c.

11.14.2.31 void ec_fsm_sii_end (ec_fsm_t * *fsm*)

SII state: END.

End state of the slave SII state machine.

Parameters:
fsm finite state machine

Definition at line 1556 of file fsm.c.

11.14.2.32 void ec_fsm_sii_error (ec_fsm_t * *fsm*)

SII state: ERROR.

End state of the slave SII state machine.

Parameters:
fsm finite state machine

Definition at line 1567 of file fsm.c.

11.14.2.33 void ec_fsm_change_start (ec_fsm_t *)

Change state: START.

Parameters:
fsm finite state machine

Definition at line 1579 of file fsm.c.

11.14.2.34 void ec_fsm_change_check (ec_fsm_t *)

Change state: CHECK.

Parameters:
fsm finite state machine

Definition at line 1597 of file fsm.c.

11.14.2.35 void ec_fsm_change_status (ec_fsm_t *)

Change state: STATUS.

Parameters:
fsm finite state machine

Definition at line 1630 of file fsm.c.

11.14.2.36 void ec_fsm_change_code (ec_fsm_t *)

Change state: CODE.

Parameters:
fsm finite state machine

Definition at line 1722 of file fsm.c.

11.14.2.37 void ec_fsm_change_ack (ec_fsm_t *)

Change state: ACK.

Parameters:
fsm finite state machine

Definition at line 1759 of file fsm.c.

11.14.2.38 void ec_fsm_change_check_ack (ec_fsm_t *)

Change state: CHECK ACK.

Parameters:
fsm finite state machine

Definition at line 1784 of file fsm.c.

11.14.2.39 void ec_fsm_change_end (ec_fsm_t *)

Change state: END.

Parameters:
fsm finite state machine

Definition at line 1825 of file fsm.c.

11.14.2.40 void ec_fsm_change_error (ec_fsm_t *)

Change state: ERROR.

Parameters:
fsm finite state machine

Definition at line 1835 of file fsm.c.

11.14.2.41 int ec_fsm_init (ec_fsm_t * *fsm*, ec_master_t * *master*)

Constructor.

Parameters:
fsm finite state machine

master EtherCAT master

Definition at line 98 of file fsm.c.

11.14.2.42 void ec_fsm_clear (ec_fsm_t * *fsm*)

Destructor.

Parameters:
fsm finite state machine

Definition at line 123 of file fsm.c.

11.14.2.43 void ec_fsm_reset (ec_fsm_t * *fsm*)

Resets the state machine.

Parameters:
fsm finite state machine

Definition at line 134 of file fsm.c.

11.14.2.44 void ec_fsm_execute (ec_fsm_t * *fsm*)

Executes the current state of the state machine.

Parameters:
fsm finite state machine

Definition at line 147 of file fsm.c.

11.14.2.45 void ec_fsm_master_action_process_states (ec_fsm_t * *fsm*)

Master action: PROC_STATES.

Processes the slave states.

Parameters:
fsm finite state machine

Definition at line 285 of file fsm.c.

11.14.2.46 void ec_fsm_master_action_next_slave_state (ec_fsm_t * *fsm*)

Master action: Get state of next slave.

Parameters:
fsm finite state machine

Definition at line 351 of file fsm.c.

11.14.2.47 void ec_fsm_master_action_addresses (ec_fsm_t * *fsm*)

Master action: ADDRESS.

Looks for slave, that have lost their configuration and writes their station address, so that they can be reconfigured later.

Parameters:
fsm finite state machine

Definition at line 489 of file fsm.c.

11.15 fsm.h File Reference

11.15.1 Detailed Description

EtherCAT finite state machines.

Definition in file **fsm.h**.

Data Structures

- struct **ec_fsm**
  Finite state machine of an EtherCAT master.

Typedefs

- typedef **ec_fsm ec_fsm_t**

Functions

- int **ec_fsm_init** (**ec_fsm_t** ∗, **ec_master_t** ∗)
  Constructor.
- void **ec_fsm_clear** (**ec_fsm_t** ∗)
  Destructor.
- void **ec_fsm_reset** (**ec_fsm_t** ∗)
  Resets the state machine.
- void **ec_fsm_execute** (**ec_fsm_t** ∗)
  Executes the current state of the state machine.

11.15.2 Typedef Documentation

11.15.2.1 typedef struct ec_fsm ec_fsm_t

See also:
ec_fsm(p. 50)

Definition at line 50 of file fsm.h.

11.16 globals.h File Reference

11.16.1 Detailed Description

Global definitions and macros.

Definition in file **globals.h**.

Data Structures

- struct **ec_code_msg_t**
  Code - Message pair.

Defines

- #define **EC_MASTER_VERSION_MAIN** 1
  master main version
- #define **EC_MASTER_VERSION_SUB** 0
  master sub version (after the dot)
- #define **EC_MASTER_VERSION_EXTRA** "stable"
  master extra version (just a string)
- #define **EC_MAX_FMMUS** 16
  maximum number of FMMUs per slave
- #define **EC_EOE_TX_QUEUE_SIZE** 100
  size of the EoE tx queue
- #define **EC_EOE_FREQUENCY** 1000
  clock frequency for the EoE state machines
- #define **EC_FRAME_HEADER_SIZE** 2
  size of an EtherCAT frame header
- #define **EC_DATAGRAM_HEADER_SIZE** 10
  size of an EtherCAT datagram header
- #define **EC_DATAGRAM_FOOTER_SIZE** 2
  size of an EtherCAT datagram footer
- #define **EC_SYNC_SIZE** 8
  size of a sync manager configuration page
- #define **EC_FMMU_SIZE** 16
  size of an FMMU configuration page

- #define **EC_MAX_DATA_SIZE**

  resulting maximum data size of a single datagram in a frame

- #define **EC_INFO**(fmt, args...) printk(KERN_INFO "EtherCAT: " fmt, ##args)

  Convenience macro for printing EtherCAT-specific information to syslog.

- #define **EC_ERR**(fmt, args...) printk(KERN_ERR "EtherCAT ERROR: " fmt, ##args)

  Convenience macro for printing EtherCAT-specific errors to syslog.

- #define **EC_WARN**(fmt, args...) printk(KERN_WARNING "EtherCAT WARNING: " fmt, ##args)

  Convenience macro for printing EtherCAT-specific warnings to syslog.

- #define **EC_DBG**(fmt, args...) printk(KERN_DEBUG "EtherCAT DEBUG: " fmt, ##args)

  Convenience macro for printing EtherCAT debug messages to syslog.

- #define **EC_LIT**(X) #X

  Helper macro for **EC_STR**()(p. 126)*, literates a macro argument.*

- #define **EC_STR**(X) EC_LIT(X)

  Converts a macro argument to a string.

- #define **EC_SYSFS_READ_ATTR**(NAME)

  Convenience macro for defining read-only SysFS attributes.

- #define **EC_SYSFS_READ_WRITE_ATTR**(NAME)

  Convenience macro for defining read-write SysFS attributes.

Functions

- void **ec_print_data** (const uint8_t *, size_t)

  Outputs frame contents for debugging purposes.

- void **ec_print_data_diff** (const uint8_t *, const uint8_t *, size_t)

  Outputs frame contents and differences for debugging purposes.

- void **ec_print_states** (uint8_t)

  Prints slave states in clear text.

11.16.2 Define Documentation

11.16.2.1 #define EC_MAX_DATA_SIZE

Value:

```
(ETH_DATA_LEN - EC_FRAME_HEADER_SIZE \
                        - EC_DATAGRAM_HEADER_SIZE - EC_DATAGRAM_FOOTER_SIZE)
```

resulting maximum data size of a single datagram in a frame

Definition at line 88 of file globals.h.

11.16.2.2 #define EC_INFO(fmt, args...) printk(KERN_INFO "EtherCAT: " fmt, ##args)

Convenience macro for printing EtherCAT-specific information to syslog.

This will print the message in *fmt* with a prefixed "EtherCAT: ".

Parameters:

fmt format string (like in printf())

args arguments (optional)

Definition at line 100 of file globals.h.

11.16.2.3 #define EC_ERR(fmt, args...) printk(KERN_ERR "EtherCAT ERROR: " fmt, ##args)

Convenience macro for printing EtherCAT-specific errors to syslog.

This will print the message in *fmt* with a prefixed "EtherCAT ERROR: ".

Parameters:

fmt format string (like in printf())

args arguments (optional)

Definition at line 110 of file globals.h.

11.16.2.4 #define EC_WARN(fmt, args...) printk(KERN_WARNING "EtherCAT WARNING: " fmt, ##args)

Convenience macro for printing EtherCAT-specific warnings to syslog.

This will print the message in *fmt* with a prefixed "EtherCAT WARNING: ".

Parameters:

fmt format string (like in printf())

args arguments (optional)

Definition at line 120 of file globals.h.

11.16.2.5 #define EC_DBG(fmt, args...) printk(KERN_DEBUG "EtherCAT DEBUG: " fmt, ##args)

Convenience macro for printing EtherCAT debug messages to syslog.

This will print the message in *fmt* with a prefixed "EtherCAT DEBUG: ".

Parameters:

fmt format string (like in printf())

args arguments (optional)

Definition at line 130 of file globals.h.

11.16.2.6 #define EC_LIT(X) #X

Helper macro for **EC_STR()**(p. 126), literates a macro argument.

Parameters:
X argument to literate.

Definition at line 138 of file globals.h.

11.16.2.7 #define EC_STR(X) EC_LIT(X)

Converts a macro argument to a string.

Parameters:
X argument to stringify.

Definition at line 145 of file globals.h.

11.16.2.8 #define EC_SYSFS_READ_ATTR(NAME)

Value:

```
static struct attribute attr_##NAME = { \
        .name = EC_STR(NAME), .owner = THIS_MODULE, .mode = S_IRUGO \
    }
```

Convenience macro for defining read-only SysFS attributes.

This results in creating a static variable called attr_*NAME*. The SysFS file will be world-readable.

Parameters:
NAME name of the attribute to create.

Definition at line 154 of file globals.h.

11.16.2.9 #define EC_SYSFS_READ_WRITE_ATTR(NAME)

Value:

```
static struct attribute attr_##NAME = { \
        .name = EC_STR(NAME), .owner = THIS_MODULE, .mode = S_IRUGO | S_IWUSR \
    }
```

Convenience macro for defining read-write SysFS attributes.

This results in creating a static variable called attr_*NAME*. The SysFS file will be word-readable plus owner-writable.

Parameters:
NAME name of the attribute to create.

Definition at line 166 of file globals.h.

11.17 mailbox.c File Reference

11.17.1 Detailed Description

Mailbox functionality.

Definition in file **mailbox.c**.

Functions

- uint8_t * **ec_slave_mbox_prepare_send** (const **ec_slave_t** *slave, **ec_datagram_t** *datagram, uint8_t type, size_t size)

  Prepares a mailbox-send datagram.

- int **ec_slave_mbox_prepare_check** (const **ec_slave_t** *slave, **ec_datagram_t** *datagram)

  Prepares a datagram for checking the mailbox state.

- int **ec_slave_mbox_check** (const **ec_datagram_t** *datagram)

  Processes a mailbox state checking datagram.

- int **ec_slave_mbox_prepare_fetch** (const **ec_slave_t** *slave, **ec_datagram_t** *datagram)

  Prepares a datagram to fetch mailbox data.

- uint8_t * **ec_slave_mbox_fetch** (const **ec_slave_t** *slave, **ec_datagram_t** *datagram, uint8_t type, size_t *size)

  Processes received mailbox data.

- uint8_t * **ec_slave_mbox_simple_io** (const **ec_slave_t** *slave, **ec_datagram_t** *datagram, size_t *size)

  Sends a mailbox datagram and waits for its reception.

- uint8_t * **ec_slave_mbox_simple_receive** (const **ec_slave_t** *slave, **ec_datagram_t** *datagram, uint8_t type, size_t *size)

  Waits for the reception of a mailbox datagram.

11.17.2 Function Documentation

11.17.2.1 uint8_t* ec_slave_mbox_prepare_send (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, uint8_t *type*, size_t *size*)

Prepares a mailbox-send datagram.

Returns:
pointer to mailbox datagram data

Parameters:
slave slave

datagram datagram

type mailbox protocol

size size of the data

Definition at line 55 of file mailbox.c.

11.17.2.2 int ec_slave_mbox_prepare_check (const ec_slave_t * *slave*, ec_datagram_t * *datagram*)

Prepares a datagram for checking the mailbox state.

Returns:
0 in case of success, else < 0

Parameters:
slave slave

datagram datagram

Definition at line 95 of file mailbox.c.

11.17.2.3 int ec_slave_mbox_check (const ec_datagram_t * *datagram*)

Processes a mailbox state checking datagram.

Returns:
0 in case of success, else < 0

Parameters:
datagram datagram

Definition at line 113 of file mailbox.c.

11.17.2.4 int ec_slave_mbox_prepare_fetch (const ec_slave_t * *slave*, ec_datagram_t * *datagram*)

Prepares a datagram to fetch mailbox data.

Returns:
0 in case of success, else < 0

Parameters:
slave slave

datagram datagram

Definition at line 125 of file mailbox.c.

11.17.2.5 uint8_t* ec_slave_mbox_fetch (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, uint8_t *type*, size_t * *size*)

Processes received mailbox data.

Returns:
pointer to the received data

Parameters:

slave slave

datagram datagram

type expected mailbox protocol

size size of the received data

Definition at line 142 of file mailbox.c.

11.17.2.6 uint8_t* ec_slave_mbox_simple_io (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, size_t * *size*)

Sends a mailbox datagram and waits for its reception.

Returns:

pointer to the received data

Parameters:

slave slave

datagram datagram

size size of the received data

Definition at line 174 of file mailbox.c.

11.17.2.7 uint8_t* ec_slave_mbox_simple_receive (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, uint8_t *type*, size_t * *size*)

Waits for the reception of a mailbox datagram.

Returns:

pointer to the received data

Parameters:

slave slave

datagram datagram

type expected protocol

size received data size

Definition at line 199 of file mailbox.c.

11.18 mailbox.h File Reference

11.18.1 Detailed Description

Mailbox functionality.

Definition in file **mailbox.h**.

Functions

- uint8_t * **ec_slave_mbox_prepare_send** (const **ec_slave_t** *, **ec_datagram_t** *, uint8_t, size_t)
  Prepares a mailbox-send datagram.
- int **ec_slave_mbox_prepare_check** (const **ec_slave_t** *, **ec_datagram_t** *)
  Prepares a datagram for checking the mailbox state.
- int **ec_slave_mbox_check** (const **ec_datagram_t** *)
  Processes a mailbox state checking datagram.
- int **ec_slave_mbox_prepare_fetch** (const **ec_slave_t** *, **ec_datagram_t** *)
  Prepares a datagram to fetch mailbox data.
- uint8_t * **ec_slave_mbox_fetch** (const **ec_slave_t** *, **ec_datagram_t** *, uint8_t, size_t *)
  Processes received mailbox data.
- uint8_t * **ec_slave_mbox_simple_io** (const **ec_slave_t** *, **ec_datagram_t** *, size_t *)
  Sends a mailbox datagram and waits for its reception.
- uint8_t * **ec_slave_mbox_simple_receive** (const **ec_slave_t** *, **ec_datagram_t** *, uint8_t, size_t *)
  Waits for the reception of a mailbox datagram.

11.18.2 Function Documentation

11.18.2.1 uint8_t* ec_slave_mbox_prepare_send (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, uint8_t *type*, size_t *size*)

Prepares a mailbox-send datagram.

Returns:
pointer to mailbox datagram data

Parameters:
slave slave
datagram datagram
type mailbox protocol
size size of the data

Definition at line 55 of file mailbox.c.

11.18.2.2 int ec_slave_mbox_prepare_check (const ec_slave_t * *slave*, ec_datagram_t * *datagram*)

Prepares a datagram for checking the mailbox state.

Returns:
0 in case of success, else < 0

Parameters:
slave slave
datagram datagram

Definition at line 95 of file mailbox.c.

11.18.2.3 int ec_slave_mbox_check (const ec_datagram_t * *datagram*)

Processes a mailbox state checking datagram.

Returns:
0 in case of success, else < 0

Parameters:
datagram datagram

Definition at line 113 of file mailbox.c.

11.18.2.4 int ec_slave_mbox_prepare_fetch (const ec_slave_t * *slave*, ec_datagram_t * *datagram*)

Prepares a datagram to fetch mailbox data.

Returns:
0 in case of success, else < 0

Parameters:
slave slave
datagram datagram

Definition at line 125 of file mailbox.c.

11.18.2.5 uint8_t* ec_slave_mbox_fetch (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, uint8_t *type*, size_t * *size*)

Processes received mailbox data.

Returns:
pointer to the received data

Parameters:
slave slave
datagram datagram
type expected mailbox protocol
size size of the received data

Definition at line 142 of file mailbox.c.

11.18.2.6 uint8_t* ec_slave_mbox_simple_io (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, size_t * *size*)

Sends a mailbox datagram and waits for its reception.

Returns:
pointer to the received data

Parameters:
slave slave

datagram datagram

size size of the received data

Definition at line 174 of file mailbox.c.

11.18.2.7 uint8_t* ec_slave_mbox_simple_receive (const ec_slave_t * *slave*, ec_datagram_t * *datagram*, uint8_t *type*, size_t * *size*)

Waits for the reception of a mailbox datagram.

Returns:
pointer to the received data

Parameters:
slave slave

datagram datagram

type expected protocol

size received data size

Definition at line 199 of file mailbox.c.

11.19 master.c File Reference

11.19.1 Detailed Description

EtherCAT master methods.

Definition in file **master.c**.

Functions

- void **ec_master_idle_run** (void *data)
  Idle mode function.
- void **ec_master_eoe_run** (unsigned long data)
  Does the Ethernet-over-EtherCAT processing.
- ssize_t **ec_show_master_attribute** (struct kobject *kobj,struct attribute *attr,char *buffer)
  Formats attribute data for SysFS read access.
- ssize_t **ec_store_master_attribute** (struct kobject *kobj,struct attribute *attr,const char *buffer,size_t size)
  Formats attribute data for SysFS write access.
- int **ec_master_init** (**ec_master_t** *master, unsigned int index, unsigned int eoeif_count)
  Master constructor.
- void **ec_master_clear** (struct kobject *kobj)
  Master destructor.
- void **ec_master_reset** (**ec_master_t** *master)
  Resets the master.
- void **ec_master_clear_slaves** (**ec_master_t** *master)
  Clears all slaves.
- void **ec_master_queue_datagram** (**ec_master_t** *master, **ec_datagram_t** *datagram)
  Places a datagram in the datagram queue.
- void **ec_master_send_datagrams** (**ec_master_t** *master)
  Sends the datagrams in the queue.
- void **ec_master_receive** (**ec_master_t** *master, const uint8_t *frame_data, size_t size)
  Processes a received frame.
- int **ec_master_simple_io** (**ec_master_t** *master, **ec_datagram_t** *datagram)
  Sends a single datagram and waits for its reception.
- int **ec_master_bus_scan** (**ec_master_t** *master)
  Scans the EtherCAT bus for slaves.

- void **ec_master_output_stats** (**ec_master_t** ∗master)
  Output statistics in cyclic mode.
- void **ec_master_idle_start** (**ec_master_t** ∗master)
  Starts the Idle mode.
- void **ec_master_idle_stop** (**ec_master_t** ∗master)
  Stops the Idle mode.
- void **ec_sync_config** (const **ec_sync_t** ∗sync, const **ec_slave_t** ∗slave, uint8_t ∗data)
  Initializes a sync manager configuration page.
- void **ec_eeprom_sync_config** (const **ec_eeprom_sync_t** ∗sync, const **ec_slave_t** ∗slave, uint8_t ∗data)
  Initializes a sync manager configuration page with EEPROM data.
- void **ec_fmmu_config** (const **ec_fmmu_t** ∗fmmu, const **ec_slave_t** ∗slave, uint8_t ∗data)
  Initializes an FMMU configuration page.
- void **ec_master_eoe_start** (**ec_master_t** ∗master)
  Starts Ethernet-over-EtherCAT processing for all EoE-capable slaves.
- void **ec_master_eoe_stop** (**ec_master_t** ∗master)
  Stops the Ethernet-over-EtherCAT processing.
- **ec_domain_t** ∗ **ecrt_master_create_domain** (**ec_master_t** ∗master)
  Creates a domain.
- int **ecrt_master_activate** (**ec_master_t** ∗master)
  Configures all slaves and leads them to the OP state.
- void **ecrt_master_deactivate** (**ec_master_t** ∗master)
  Resets all slaves to INIT state.
- int **ecrt_master_fetch_sdo_lists** (**ec_master_t** ∗master)
  Fetches the SDO dictionaries of all slaves.
- void **ecrt_master_sync_io** (**ec_master_t** ∗master)
  Sends queued datagrams and waits for their reception.
- void **ecrt_master_async_send** (**ec_master_t** ∗master)
  Asynchronous sending of datagrams.
- void **ecrt_master_async_receive** (**ec_master_t** ∗master)
  Asynchronous receiving of datagrams.
- void **ecrt_master_prepare_async_io** (**ec_master_t** ∗master)
  Prepares synchronous IO.
- void **ecrt_master_run** (**ec_master_t** ∗master)

Does all cyclic master work.

- **ec_slave_t** ∗ **ecrt_master_get_slave** (const **ec_master_t** ∗master, const char ∗address)

  Translates an ASCII coded bus-address to a slave pointer.

- void **ecrt_master_callbacks** (**ec_master_t** ∗master, int(∗request_cb)(void ∗), void(∗release_cb)(void ∗), void ∗cb_data)

  Sets the locking callbacks.

- int **ecrt_master_start_eoe** (**ec_master_t** ∗master)

  Starts Ethernet-over-EtherCAT processing for all EoE-capable slaves.

- void **ecrt_master_debug** (**ec_master_t** ∗master, int level)

  Sets the debug level of the master.

- void **ecrt_master_print** (const **ec_master_t** ∗master, unsigned int verbosity)

  Outputs all master information.

11.19.2 Function Documentation

11.19.2.1 void ec_master_idle_run (void ∗)

Idle mode function.

Parameters:
data master pointer

Definition at line 727 of file master.c.

11.19.2.2 void ec_master_eoe_run (unsigned *long*)

Does the Ethernet-over-EtherCAT processing.

Parameters:
data master pointer

Definition at line 1017 of file master.c.

11.19.2.3 ssize_t ec_show_master_attribute (struct kobject ∗ *kobj*, struct attribute ∗ *attr*, char ∗ *buffer*)

Formats attribute data for SysFS read access.

Returns:
number of bytes to read

Parameters:
kobj kobject

attr attribute

buffer memory to store data

Definition at line 836 of file master.c.

11.19.2.4 ssize_t ec_store_master_attribute (struct kobject * *kobj*, struct attribute * *attr*, const char * *buffer*, size_t *size*)

Formats attribute data for SysFS write access.

Returns:
number of bytes processed, or negative error code

Parameters:
kobj slave's kobject

attr attribute

buffer memory with data

size size of data to store

Definition at line 873 of file master.c.

11.19.2.5 int ec_master_init (ec_master_t * *master*, unsigned int *index*, unsigned int *eoeif_count*)

Master constructor.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

index master index

eoeif_count number of EoE interfaces

Definition at line 101 of file master.c.

11.19.2.6 void ec_master_clear (struct kobject * *kobj*)

Master destructor.

Removes all pending datagrams, clears the slave list, clears all domains and frees the device.

Parameters:
kobj kobject of the master

Definition at line 180 of file master.c.

11.19.2.7 void ec_master_reset (ec_master_t * *master*)

Resets the master.

Note: This function has to be called, everytime ec_master_release() is called, to free the slave list, domains etc.

Parameters:
master EtherCAT master

Definition at line 215 of file master.c.

11.19.2.8 void ec_master_queue_datagram (ec_master_t * *master*, ec_datagram_t * *datagram*)

Places a datagram in the datagram queue.

Parameters:
master EtherCAT master

datagram datagram

Definition at line 283 of file master.c.

11.19.2.9 void ec_master_send_datagrams (ec_master_t * *master*)

Sends the datagrams in the queue.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

Definition at line 310 of file master.c.

11.19.2.10 void ec_master_receive (ec_master_t * *master*, const uint8_t * *frame_data*, size_t *size*)

Processes a received frame.

This function is called by the network driver for every received frame.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

frame_data received data

size size of the received data

Definition at line 411 of file master.c.

11.19.2.11 int ec_master_simple_io (ec_master_t * *master*, ec_datagram_t * *datagram*)

Sends a single datagram and waits for its reception.

If the slave doesn't respond, the datagram is sent again.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

datagram datagram

Definition at line 498 of file master.c.

11.19.2.12 int ec_master_bus_scan (ec_master_t * *master*)

Scans the EtherCAT bus for slaves.

Creates a list of slave structures for further processing.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

Definition at line 542 of file master.c.

11.19.2.13 void ec_master_output_stats (ec_master_t * *master*)

Output statistics in cyclic mode.

This function outputs statistical data on demand, but not more often than necessary. The output happens at most once a second.

Parameters:
master EtherCAT master

Definition at line 650 of file master.c.

11.19.2.14 void ec_master_idle_start (ec_master_t * *master*)

Starts the Idle mode.

Parameters:
master EtherCAT master

Definition at line 681 of file master.c.

11.19.2.15 void ec_master_idle_stop (ec_master_t * *master*)

Stops the Idle mode.

Parameters:
master EtherCAT master

Definition at line 703 of file master.c.

11.19.2.16 void ec_sync_config (const ec_sync_t * *sync*, const ec_slave_t * *slave*, uint8_t * *data*)

Initializes a sync manager configuration page.

The referenced memory (*data*) must be at least EC_SYNC_SIZE bytes.

Parameters:
sync sync manager

slave EtherCAT slave

data > configuration memory

Definition at line 755 of file master.c.

11.19.2.17 void ec_eeprom_sync_config (const ec_eeprom_sync_t * *sync*, const ec_slave_t * *slave*, uint8_t * *data*)

Initializes a sync manager configuration page with EEPROM data.

The referenced memory (*data*) must be at least EC_SYNC_SIZE bytes.

Parameters:
sync sync manager

slave EtherCAT slave

data > configuration memory

Definition at line 778 of file master.c.

11.19.2.18 void ec_fmmu_config (const ec_fmmu_t * *fmmu*, const ec_slave_t * *slave*, uint8_t * *data*)

Initializes an FMMU configuration page.

The referenced memory (*data*) must be at least EC_FMMU_SIZE bytes.

Parameters:
fmmu FMMU

slave EtherCAT slave

data > configuration memory

Definition at line 809 of file master.c.

11.19.2.19 void ec_master_eoe_start (ec_master_t * *master*)

Starts Ethernet-over-EtherCAT processing for all EoE-capable slaves.

Parameters:
master EtherCAT master

Definition at line 928 of file master.c.

11.19.2.20 void ec_master_eoe_stop (ec_master_t * *master*)

Stops the Ethernet-over-EtherCAT processing.

Parameters:
master EtherCAT master

Definition at line 990 of file master.c.

11.20 master.h File Reference

11.20.1 Detailed Description

EtherCAT master structure.

Definition in file **master.h**.

Data Structures

- struct **ec_stats_t**
  Cyclic statistics.
- struct **ec_master**
  EtherCAT master.

Enumerations

- enum **ec_master_mode_t** { **EC_MASTER_MODE_ORPHANED**, **EC_MASTER_MODE_IDLE**, **EC_MASTER_MODE_RUNNING** }
  EtherCAT master mode.

Functions

- int **ec_master_init** (**ec_master_t** *, unsigned int, unsigned int)
  Master constructor.
- void **ec_master_clear** (struct kobject *)
  Master destructor.
- void **ec_master_reset** (**ec_master_t** *)
  Resets the master.
- void **ec_master_idle_start** (**ec_master_t** *)
  Starts the Idle mode.
- void **ec_master_idle_stop** (**ec_master_t** *)
  Stops the Idle mode.
- void **ec_master_eoe_start** (**ec_master_t** *)
  Starts Ethernet-over-EtherCAT processing for all EoE-capable slaves.
- void **ec_master_eoe_stop** (**ec_master_t** *)
  Stops the Ethernet-over-EtherCAT processing.
- void **ec_master_receive** (**ec_master_t** *, const uint8_t *, size_t)
  Processes a received frame.

- void **ec_master_queue_datagram** (**ec_master_t** ∗, **ec_datagram_t** ∗)
  Places a datagram in the datagram queue.
- int **ec_master_simple_io** (**ec_master_t** ∗, **ec_datagram_t** ∗)
  Sends a single datagram and waits for its reception.
- int **ec_master_bus_scan** (**ec_master_t** ∗)
  Scans the EtherCAT bus for slaves.
- void **ec_master_clear_slaves** (**ec_master_t** ∗)
  Clears all slaves.
- void **ec_sync_config** (const **ec_sync_t** ∗, const **ec_slave_t** ∗, uint8_t ∗)
  Initializes a sync manager configuration page.
- void **ec_eeprom_sync_config** (const **ec_eeprom_sync_t** ∗, const **ec_slave_t** ∗, uint8_t ∗)
  Initializes a sync manager configuration page with EEPROM data.
- void **ec_fmmu_config** (const **ec_fmmu_t** ∗, const **ec_slave_t** ∗, uint8_t ∗)
  Initializes an FMMU configuration page.
- void **ec_master_output_stats** (**ec_master_t** ∗)
  Output statistics in cyclic mode.

11.20.2 Function Documentation

11.20.2.1 int ec_master_init (ec_master_t ∗ *master*, unsigned int *index*, unsigned int *eoeif_count*)

Master constructor.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master
index master index
eoeif_count number of EoE interfaces

Definition at line 101 of file master.c.

11.20.2.2 void ec_master_clear (struct kobject ∗ *kobj*)

Master destructor.

Removes all pending datagrams, clears the slave list, clears all domains and frees the device.

Parameters:
kobj kobject of the master

Definition at line 180 of file master.c.

11.20.2.3 void ec_master_reset (ec_master_t * *master*)

Resets the master.

Note: This function has to be called, everytime ec_master_release() is called, to free the slave list, domains etc.

Parameters:
master EtherCAT master

Definition at line 215 of file master.c.

11.20.2.4 void ec_master_receive (ec_master_t * *master*, const uint8_t * *frame_data*, size_t *size*)

Processes a received frame.

This function is called by the network driver for every received frame.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master
frame_data received data
size size of the received data

Definition at line 411 of file master.c.

11.20.2.5 int ec_master_simple_io (ec_master_t * *master*, ec_datagram_t * *datagram*)

Sends a single datagram and waits for its reception.

If the slave doesn't respond, the datagram is sent again.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master
datagram datagram

Definition at line 498 of file master.c.

11.20.2.6 int ec_master_bus_scan (ec_master_t * *master*)

Scans the EtherCAT bus for slaves.

Creates a list of slave structures for further processing.

Returns:
0 in case of success, else < 0

Parameters:
master EtherCAT master

Definition at line 542 of file master.c.

11.20.2.7 void ec_sync_config (const ec_sync_t * *sync*, const ec_slave_t * *slave*, uint8_t * *data*)

Initializes a sync manager configuration page.

The referenced memory (*data*) must be at least EC_SYNC_SIZE bytes.

Parameters:

sync sync manager

slave EtherCAT slave

data > configuration memory

Definition at line 755 of file master.c.

11.20.2.8 void ec_eeprom_sync_config (const ec_eeprom_sync_t * *sync*, const ec_slave_t * *slave*, uint8_t * *data*)

Initializes a sync manager configuration page with EEPROM data.

The referenced memory (*data*) must be at least EC_SYNC_SIZE bytes.

Parameters:

sync sync manager

slave EtherCAT slave

data > configuration memory

Definition at line 778 of file master.c.

11.20.2.9 void ec_fmmu_config (const ec_fmmu_t * *fmmu*, const ec_slave_t * *slave*, uint8_t * *data*)

Initializes an FMMU configuration page.

The referenced memory (*data*) must be at least EC_FMMU_SIZE bytes.

Parameters:

fmmu FMMU

slave EtherCAT slave

data > configuration memory

Definition at line 809 of file master.c.

11.20.2.10 void ec_master_output_stats (ec_master_t * *master*)

Output statistics in cyclic mode.

This function outputs statistical data on demand, but not more often than necessary. The output happens at most once a second.

Parameters:

master EtherCAT master

Definition at line 650 of file master.c.

11.21 module.c File Reference

11.21.1 Detailed Description

EtherCAT master driver module.

Definition in file **module.c**.

Defines

- #define **COMPILE_INFO**
  Compile version info.

Functions

- int __init **ec_init_module** (void)
  Module initialization.
- void __exit **ec_cleanup_module** (void)
  Module cleanup.
- **ec_master_t** * **ec_find_master** (unsigned int master_index)
  Gets a handle to a certain master.
- void **ec_print_data** (const uint8_t *data, size_t size)
  Outputs frame contents for debugging purposes.
- void **ec_print_data_diff** (const uint8_t *d1, const uint8_t *d2, size_t size)
  Outputs frame contents and differences for debugging purposes.
- void **ec_print_states** (const uint8_t states)
  Prints slave states in clear text.
- **ec_device_t** * **ecdev_register** (unsigned int master_index, struct net_device *net_dev, **ec_isr_t** isr, struct module *module)
  Connects an EtherCAT device to a certain master.
- void **ecdev_unregister** (unsigned int master_index, **ec_device_t** *device)
  Disconnect an EtherCAT device from the master.
- int **ecdev_start** (unsigned int master_index)
  Starts the master associated with the device.
- void **ecdev_stop** (unsigned int master_index)
  Stops the master associated with the device.
- **ec_master_t** * **ecrt_request_master** (unsigned int master_index)
  Reserves an EtherCAT master for realtime operation.

- void **ecrt_release_master** (**ec_master_t** *master)
  Releases a reserved EtherCAT master.

Variables

- static int **ec_master_count** = 1
  parameter value, number of masters
- static int **ec_eoeif_count** = 0
  parameter value, number of EoE interf.
- static struct list_head **ec_masters**
  list of masters

11.21.2 Define Documentation

11.21.2.1 #define COMPILE_INFO

Value:

```
EC_STR(EC_MASTER_VERSION_MAIN) \
                      "." EC_STR(EC_MASTER_VERSION_SUB) \
                      " (" EC_MASTER_VERSION_EXTRA ")" \
                      " - rev. " EC_STR(SVNREV) \
                      ", compiled by " EC_STR(USER) \
                      " at " __DATE__ " " __TIME__
```

Compile version info.

Definition at line 60 of file module.c.

11.21.3 Function Documentation

11.21.3.1 int __init ec_init_module (void)

Module initialization.

Initializes *ec_master_count* masters.

Returns:
0 on success, else < 0

Definition at line 97 of file module.c.

11.21.3.2 void __exit ec_cleanup_module (void)

Module cleanup.

Clears all master instances.

Definition at line 152 of file module.c.

11.21.3.3 ec_master_t* ec_find_master (unsigned int *master_index*)

Gets a handle to a certain master.

Returns:
pointer to master

Parameters:
master_index master index

Definition at line 174 of file module.c.

11.21.3.4 void ec_print_data (const uint8_t * *data*, size_t *size*)

Outputs frame contents for debugging purposes.

Parameters:
data pointer to data
size number of bytes to output

Definition at line 192 of file module.c.

11.21.3.5 void ec_print_data_diff (const uint8_t * *d1*, const uint8_t * *d2*, size_t *size*)

Outputs frame contents and differences for debugging purposes.

Parameters:
d1 first data
d2 second data
size number of bytes to output

Definition at line 215 of file module.c.

11.21.3.6 void ec_print_states (const uint8_t *states*)

Prints slave states in clear text.

Parameters:
states slave states

Definition at line 240 of file module.c.

11.22 slave.c File Reference

11.22.1 Detailed Description

EtherCAT slave methods.

Definition in file **slave.c**.

Functions

- int **ec_slave_fetch_categories** (**ec_slave_t** ∗slave)
  Fetches data from slave's EEPROM.
- ssize_t **ec_show_slave_attribute** (struct kobject ∗kobj,struct attribute ∗attr,char ∗buffer)
  Formats attribute data for SysFS read access.
- ssize_t **ec_store_slave_attribute** (struct kobject ∗kobj,struct attribute ∗attr,const char ∗buffer,size_t size)
  Formats attribute data for SysFS write access.
- int **ec_slave_init** (**ec_slave_t** ∗slave, **ec_master_t** ∗master, uint16_t ring_position, uint16_t station_-address)
  Slave constructor.
- void **ec_slave_clear** (struct kobject ∗kobj)
  Slave destructor.
- int **ec_slave_fetch** (**ec_slave_t** ∗slave)
  Reads all necessary information from a slave.
- int **ec_slave_sii_read16** (**ec_slave_t** ∗slave, uint16_t offset, uint16_t ∗target)
  Reads 16 bit from the slave information interface (SII).
- int **ec_slave_sii_read32** (**ec_slave_t** ∗slave, uint16_t offset, uint32_t ∗target)
  Reads 32 bit from the slave information interface (SII).
- int **ec_slave_sii_write16** (**ec_slave_t** ∗slave, uint16_t offset, uint16_t value)
  Writes 16 bit of data to the slave information interface (SII).
- int **ec_slave_fetch_strings** (**ec_slave_t** ∗slave, const uint8_t ∗data)
  Fetches data from a STRING category.
- int **ec_slave_fetch_general** (**ec_slave_t** ∗slave, const uint8_t ∗data)
  Fetches data from a GENERAL category.
- int **ec_slave_fetch_sync** (**ec_slave_t** ∗slave, const uint8_t ∗data, size_t word_count)
  Fetches data from a SYNC MANAGER category.
- int **ec_slave_fetch_pdo** (**ec_slave_t** ∗slave, const uint8_t ∗data, size_t word_count, **ec_pdo_type_t** pdo_type)

Fetches data from a [RT]XPDO category.

- int **ec_slave_locate_string** (**ec_slave_t** ∗slave, unsigned int index, char ∗∗ptr)
  Searches the string list for an index and allocates a new string.
- void **ec_slave_state_ack** (**ec_slave_t** ∗slave, uint8_t state)
  Acknowledges an error after a state transition.
- void **ec_slave_read_al_status_code** (**ec_slave_t** ∗slave)
  Reads the AL status code of a slave and displays it.
- int **ec_slave_state_change** (**ec_slave_t** ∗slave, uint8_t state)
  Does a state transition.
- int **ec_slave_prepare_fmmu** (**ec_slave_t** ∗slave, const **ec_domain_t** ∗domain, const **ec_sync_t** ∗sync)
  Prepares an FMMU configuration.
- void **ec_slave_print** (const **ec_slave_t** ∗slave, unsigned int verbosity)
  Outputs all information about a certain slave.
- int **ec_slave_check_crc** (**ec_slave_t** ∗slave)
  Outputs the values of the CRC faoult counters and resets them.
- ssize_t **ec_slave_write_eeprom** (**ec_slave_t** ∗slave, const uint8_t ∗data, size_t size)
  Schedules an EEPROM write operation.
- size_t **ec_slave_calc_sync_size** (const **ec_slave_t** ∗slave, const **ec_sync_t** ∗sync)
- uint16_t **ec_slave_calc_eeprom_sync_size** (const **ec_slave_t** ∗slave, const **ec_eeprom_sync_t** ∗sync)
  Calculates the size of a sync manager by evaluating PDO sizes.
- int **ecrt_slave_write_alias** (**ec_slave_t** ∗slave, uint16_t alias)
  Writes the "configured station alias" to the slave's EEPROM.
- int **ecrt_slave_field_size** (**ec_slave_t** ∗slave, const char ∗field_name, unsigned int field_index, size_t size)

Variables

- const **ec_code_msg_t al_status_messages** [ ]
  Application layer status messages.

11.22.2 Function Documentation

11.22.2.1 int ec_slave_fetch_categories (ec_slave_t ∗ *slave*)

Fetches data from slave's EEPROM.

Returns:
0 in case of success, else < 0

Todo
memory allocation

Parameters:
slave EtherCAT slave

Definition at line 533 of file slave.c.

11.22.2.2 ssize_t ec_show_slave_attribute (struct kobject * *kobj*, struct attribute * *attr*, char * *buffer*)

Formats attribute data for SysFS read access.

Returns:
number of bytes to read

Parameters:
kobj slave's kobject

attr attribute

buffer memory to store data

Definition at line 1346 of file slave.c.

11.22.2.3 ssize_t ec_store_slave_attribute (struct kobject * *kobj*, struct attribute * *attr*, const char * *buffer*, size_t *size*)

Formats attribute data for SysFS write access.

Returns:
number of bytes processed, or negative error code

Parameters:
kobj slave's kobject

attr attribute

buffer memory with data

size size of data to store

Definition at line 1424 of file slave.c.

11.22.2.4 int ec_slave_init (ec_slave_t * *slave*, ec_master_t * *master*, uint16_t *ring_position*, uint16_t *station_address*)

Slave constructor.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
master EtherCAT master
ring_position ring position
station_address station address to configure

Definition at line 107 of file slave.c.

11.22.2.5 void ec_slave_clear (struct kobject * *kobj*)

Slave destructor.

Parameters:
kobj kobject of the slave

Definition at line 185 of file slave.c.

11.22.2.6 int ec_slave_fetch (ec_slave_t * *slave*)

Reads all necessary information from a slave.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

Definition at line 260 of file slave.c.

11.22.2.7 int ec_slave_sii_read16 (ec_slave_t * *slave*, uint16_t *offset*, uint16_t * *target*)

Reads 16 bit from the slave information interface (SII).

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
offset address of the SII register to read
target target memory

Definition at line 339 of file slave.c.

11.22.2.8 int ec_slave_sii_read32 (ec_slave_t * *slave*, uint16_t *offset*, uint32_t * *target*)

Reads 32 bit from the slave information interface (SII).

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

offset address of the SII register to read

target target memory

Definition at line 400 of file slave.c.

11.22.2.9 int ec_slave_sii_write16 (ec_slave_t * *slave*, uint16_t *offset*, uint16_t *value*)

Writes 16 bit of data to the slave information interface (SII).

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

offset address of the SII register to write

value new value

Definition at line 461 of file slave.c.

11.22.2.10 int ec_slave_fetch_strings (ec_slave_t * *slave*, const uint8_t * *data*)

Fetches data from a STRING category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

data category data

Definition at line 625 of file slave.c.

11.22.2.11 int ec_slave_fetch_general (ec_slave_t * *slave*, const uint8_t * *data*)

Fetches data from a GENERAL category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

data category data

Definition at line 663 of file slave.c.

11.22.2.12 int ec_slave_fetch_sync (ec_slave_t * *slave*, const uint8_t * *data*, size_t *word_count*)

Fetches data from a SYNC MANAGER category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

data category data

word_count number of words

Definition at line 692 of file slave.c.

11.22.2.13 int ec_slave_fetch_pdo (ec_slave_t * *slave*, const uint8_t * *data*, size_t *word_count*, ec_pdo_type_t *pdo_type*)

Fetches data from a [RT]XPDO category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

data category data

word_count number of words

pdo_type PDO type

Definition at line 728 of file slave.c.

11.22.2.14 int ec_slave_locate_string (ec_slave_t * *slave*, unsigned int *index*, char ** *ptr*)

Searches the string list for an index and allocates a new string.

Returns:
0 in case of success, else < 0

Todo
documentation

Parameters:
slave EtherCAT slave

index string index

ptr Address of the string pointer

Definition at line 790 of file slave.c.

11.22.2.15 void ec_slave_state_ack (ec_slave_t * *slave*, uint8_t *state*)

Acknowledges an error after a state transition.

Parameters:
slave EtherCAT slave
state previous state

Definition at line 838 of file slave.c.

11.22.2.16 void ec_slave_read_al_status_code (ec_slave_t * *slave*)

Reads the AL status code of a slave and displays it.

If the AL status code is not supported, or if no error occurred (both resulting in code = 0), nothing is displayed.

Parameters:
slave EtherCAT slave

Definition at line 896 of file slave.c.

11.22.2.17 int ec_slave_state_change (ec_slave_t * *slave*, uint8_t *state*)

Does a state transition.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
state new state

Definition at line 931 of file slave.c.

11.22.2.18 int ec_slave_prepare_fmmu (ec_slave_t * *slave*, const ec_domain_t * *domain*, const ec_sync_t * *sync*)

Prepares an FMMU configuration.

Configuration data for the FMMU is saved in the slave structure and is written to the slave in **ecrt_master_activate()**(p. 21). The FMMU configuration is done in a way, that the complete data range of the corresponding sync manager is covered. Seperate FMMUs arce configured for each domain. If the FMMU configuration is already prepared, the function returns with success.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
domain domain
sync sync manager

Definition at line 1009 of file slave.c.

11.22.2.19 void ec_slave_print (const ec_slave_t * *slave*, unsigned int *verbosity*)

Outputs all information about a certain slave.

Verbosity:

- 0: Only slave types and addresses
- 1: with EEPROM information
- >1: with SDO dictionaries

Parameters:
slave EtherCAT slave
verbosity verbosity level

Definition at line 1047 of file slave.c.

11.22.2.20 int ec_slave_check_crc (ec_slave_t * *slave*)

Outputs the values of the CRC faoult counters and resets them.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

Definition at line 1212 of file slave.c.

11.22.2.21 ssize_t ec_slave_write_eeprom (ec_slave_t * *slave*, const uint8_t * *data*, size_t *size*)

Schedules an EEPROM write operation.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
data new EEPROM data
size size of data in bytes

Definition at line 1269 of file slave.c.

11.22.2.22 size_t ec_slave_calc_sync_size (const ec_slave_t * *slave*, const ec_sync_t * *sync*)

Returns:
size of sync manager contents

Parameters:
slave EtherCAT slave
sync sync manager

Definition at line 1465 of file slave.c.

11.22.2.23 uint16_t ec_slave_calc_eeprom_sync_size (const ec_slave_t * *slave*, const ec_eeprom_sync_t * *sync*)

Calculates the size of a sync manager by evaluating PDO sizes.

Returns:
sync manager size

Parameters:
slave EtherCAT slave

sync sync manager

Definition at line 1503 of file slave.c.

11.23 slave.h File Reference

11.23.1 Detailed Description

EtherCAT stave structure.

Definition in file **slave.h**.

Data Structures

- struct **ec_fmmu_t**
  FMMU configuration.
- struct **ec_eeprom_string_t**
  String object (EEPROM).
- struct **ec_eeprom_sync_t**
  Sync manager configuration (EEPROM).
- struct **ec_eeprom_pdo_t**
  PDO description (EEPROM).
- struct **ec_eeprom_pdo_entry_t**
  PDO entry description (EEPROM).
- struct **ec_sdo_t**
  CANopen SDO.
- struct **ec_sdo_entry_t**
  CANopen SDO entry.
- struct **ec_varsize_t**
  Variable-sized field information.
- struct **ec_slave**
  EtherCAT slave.

Enumerations

- enum **ec_slave_state_t** {
  EC_SLAVE_STATE_UNKNOWN = 0x00, **EC_SLAVE_STATE_INIT** = 0x01, **EC_SLAVE_-STATE_PREOP** = 0x02, **EC_SLAVE_STATE_SAVEOP** = 0x04,
  EC_SLAVE_STATE_OP = 0x08, **EC_ACK** = 0x10 }
  State of an EtherCAT slave.
- enum {
  EC_MBOX_AOE = 0x01, **EC_MBOX_EOE** = 0x02, **EC_MBOX_COE** = 0x04, **EC_MBOX_-FOE** = 0x08,
  EC_MBOX_SOE = 0x10, **EC_MBOX_VOE** = 0x20 }

Supported mailbox protocols.

- enum **ec_pdo_type_t** { **EC_RX_PDO**, **EC_TX_PDO** }
  PDO type.

Functions

- int **ec_slave_init** (**ec_slave_t** *, **ec_master_t** *, uint16_t, uint16_t)
  Slave constructor.
- void **ec_slave_clear** (struct kobject *)
  Slave destructor.
- int **ec_slave_fetch** (**ec_slave_t** *)
  Reads all necessary information from a slave.
- int **ec_slave_sii_read16** (**ec_slave_t** *, uint16_t, uint16_t *)
  Reads 16 bit from the slave information interface (SII).
- int **ec_slave_sii_read32** (**ec_slave_t** *, uint16_t, uint32_t *)
  Reads 32 bit from the slave information interface (SII).
- int **ec_slave_sii_write16** (**ec_slave_t** *, uint16_t, uint16_t)
  Writes 16 bit of data to the slave information interface (SII).
- int **ec_slave_state_change** (**ec_slave_t** *, uint8_t)
  Does a state transition.
- int **ec_slave_prepare_fmmu** (**ec_slave_t** *, const **ec_domain_t** *, const **ec_sync_t** *)
  Prepares an FMMU configuration.
- int **ec_slave_fetch_sdo_list** (**ec_slave_t** *)
  Fetches the SDO dictionary of a slave.
- int **ec_slave_fetch_strings** (**ec_slave_t** *, const uint8_t *)
  Fetches data from a STRING category.
- int **ec_slave_fetch_general** (**ec_slave_t** *, const uint8_t *)
  Fetches data from a GENERAL category.
- int **ec_slave_fetch_sync** (**ec_slave_t** *, const uint8_t *, size_t)
  Fetches data from a SYNC MANAGER category.
- int **ec_slave_fetch_pdo** (**ec_slave_t** *, const uint8_t *, size_t, **ec_pdo_type_t**)
  Fetches data from a [RT]XPDO category.
- int **ec_slave_locate_string** (**ec_slave_t** *, unsigned int, char **)
  Searches the string list for an index and allocates a new string.

- size_t **ec_slave_calc_sync_size** (const **ec_slave_t** *, const **ec_sync_t** *)
- uint16_t **ec_slave_calc_eeprom_sync_size** (const **ec_slave_t** *, const **ec_eeprom_sync_t** *)
  Calculates the size of a sync manager by evaluating PDO sizes.
- void **ec_slave_print** (const **ec_slave_t** *, unsigned int)
  Outputs all information about a certain slave.
- int **ec_slave_check_crc** (**ec_slave_t** *)
  Outputs the values of the CRC faoult counters and resets them.

11.23.2 Enumeration Type Documentation

11.23.2.1 enum ec_slave_state_t

State of an EtherCAT slave.

Enumerator:

EC_SLAVE_STATE_UNKNOWN unknown state

EC_SLAVE_STATE_INIT INIT state (no mailbox communication, no IO).

EC_SLAVE_STATE_PREOP PREOP state (mailbox communication, no IO).

EC_SLAVE_STATE_SAVEOP SAVEOP (mailbox communication and input update).

EC_SLAVE_STATE_OP OP (mailbox communication and input/output update).

EC_ACK Acknoledge bit (no state).

Definition at line 57 of file slave.h.

11.23.2.2 anonymous enum

Supported mailbox protocols.

Enumerator:

EC_MBOX_AOE ADS-over-EtherCAT.

EC_MBOX_EOE Ethernet-over-EtherCAT.

EC_MBOX_COE CANopen-over-EtherCAT.

EC_MBOX_FOE File-Access-over-EtherCAT.

EC_MBOX_SOE Servo-Profile-over-EtherCAT.

EC_MBOX_VOE Vendor specific.

Definition at line 80 of file slave.h.

11.23.2.3 enum ec_pdo_type_t

PDO type.

Enumerator:

EC_RX_PDO Reveive PDO.

EC_TX_PDO Transmit PDO.

Definition at line 141 of file slave.h.

11.23.3 Function Documentation

11.23.3.1 int ec_slave_init (ec_slave_t * *slave*, ec_master_t * *master*, uint16_t *ring_position*, uint16_t *station_address*)

Slave constructor.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
master EtherCAT master
ring_position ring position
station_address station address to configure

Definition at line 107 of file slave.c.

11.23.3.2 int ec_slave_fetch (ec_slave_t * *slave*)

Reads all necessary information from a slave.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

Definition at line 260 of file slave.c.

11.23.3.3 int ec_slave_sii_read16 (ec_slave_t * *slave*, uint16_t *offset*, uint16_t * *target*)

Reads 16 bit from the slave information interface (SII).

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
offset address of the SII register to read
target target memory

Definition at line 339 of file slave.c.

11.23.3.4 int ec_slave_sii_read32 (ec_slave_t * *slave*, uint16_t *offset*, uint32_t * *target*)

Reads 32 bit from the slave information interface (SII).

Returns:
0 in case of success, else < 0

Parameters:

slave EtherCAT slave

offset address of the SII register to read

target target memory

Definition at line 400 of file slave.c.

11.23.3.5 int ec_slave_sii_write16 (ec_slave_t * *slave*, uint16_t *offset*, uint16_t *value*)

Writes 16 bit of data to the slave information interface (SII).

Returns:

0 in case of success, else < 0

Parameters:

slave EtherCAT slave

offset address of the SII register to write

value new value

Definition at line 461 of file slave.c.

11.23.3.6 int ec_slave_state_change (ec_slave_t * *slave*, uint8_t *state*)

Does a state transition.

Returns:

0 in case of success, else < 0

Parameters:

slave EtherCAT slave

state new state

Definition at line 931 of file slave.c.

11.23.3.7 int ec_slave_prepare_fmmu (ec_slave_t * *slave*, const ec_domain_t * *domain*, const ec_sync_t * *sync*)

Prepares an FMMU configuration.

Configuration data for the FMMU is saved in the slave structure and is written to the slave in **ecrt_master_activate()**(p. 21). The FMMU configuration is done in a way, that the complete data range of the corresponding sync manager is covered. Seperate FMMUs arce configured for each domain. If the FMMU configuration is already prepared, the function returns with success.

Returns:

0 in case of success, else < 0

Parameters:

slave EtherCAT slave

domain domain

sync sync manager

Definition at line 1009 of file slave.c.

11.23.3.8 int ec_slave_fetch_sdo_list (ec_slave_t * *slave*)

Fetches the SDO dictionary of a slave.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

Definition at line 270 of file canopen.c.

11.23.3.9 int ec_slave_fetch_strings (ec_slave_t * *slave*, const uint8_t * *data*)

Fetches data from a STRING category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
data category data

Definition at line 625 of file slave.c.

11.23.3.10 int ec_slave_fetch_general (ec_slave_t * *slave*, const uint8_t * *data*)

Fetches data from a GENERAL category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
data category data

Definition at line 663 of file slave.c.

11.23.3.11 int ec_slave_fetch_sync (ec_slave_t * *slave*, const uint8_t * *data*, size_t *word_count*)

Fetches data from a SYNC MANAGER category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
data category data
word_count number of words

Definition at line 692 of file slave.c.

11.23.3.12 int ec_slave_fetch_pdo (ec_slave_t * *slave*, const uint8_t * *data*, size_t *word_count*, ec_pdo_type_t *pdo_type*)

Fetches data from a [RT]XPDO category.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

data category data

word_count number of words

pdo_type PDO type

Definition at line 728 of file slave.c.

11.23.3.13 int ec_slave_locate_string (ec_slave_t * *slave*, unsigned int *index*, char ** *ptr*)

Searches the string list for an index and allocates a new string.

Returns:
0 in case of success, else < 0

Todo
documentation

Parameters:
slave EtherCAT slave

index string index

ptr Address of the string pointer

Definition at line 790 of file slave.c.

11.23.3.14 size_t ec_slave_calc_sync_size (const ec_slave_t * *slave*, const ec_sync_t * *sync*)

Returns:
size of sync manager contents

Parameters:
slave EtherCAT slave

sync sync manager

Definition at line 1465 of file slave.c.

11.23.3.15 uint16_t ec_slave_calc_eeprom_sync_size (const ec_slave_t * *slave*, const ec_eeprom_sync_t * *sync*)

Calculates the size of a sync manager by evaluating PDO sizes.

Returns:
sync manager size

Parameters:
slave EtherCAT slave
sync sync manager

Definition at line 1503 of file slave.c.

11.23.3.16 void ec_slave_print (const ec_slave_t * *slave*, unsigned int *verbosity*)

Outputs all information about a certain slave.

Verbosity:

- 0: Only slave types and addresses
- 1: with EEPROM information
- >1: with SDO dictionaries

Parameters:
slave EtherCAT slave
verbosity verbosity level

Definition at line 1047 of file slave.c.

11.23.3.17 int ec_slave_check_crc (ec_slave_t * *slave*)

Outputs the values of the CRC faoult counters and resets them.

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

Definition at line 1212 of file slave.c.

11.24 types.c File Reference

11.24.1 Detailed Description

EtherCAT slave descriptions.

Definition in file **types.c**.

Variables

- **ec_slave_ident_t slave_idents** [ ]
  array with slave descriptions

11.24.2 Variable Documentation

11.24.2.1 ec_slave_ident_t slave_idents[ ]

Initial value:

```
{
    {0x00000002, 0x03EC3052, &Beckhoff_EL1004},
    {0x00000002, 0x03F63052, &Beckhoff_EL1014},
    {0x00000002, 0x044C2C52, &Beckhoff_EK1100},
    {0x00000002, 0x04562C52, &Beckhoff_EK1110},
    {0x00000002, 0x04602C22, &Beckhoff_BK1120},
    {0x00000002, 0x07D43052, &Beckhoff_EL2004},
    {0x00000002, 0x07F03052, &Beckhoff_EL2032},
    {0x00000002, 0x0C1E3052, &Beckhoff_EL3102},
    {0x00000002, 0x0C5A3052, &Beckhoff_EL3162},
    {0x00000002, 0x10063052, &Beckhoff_EL4102},
    {0x00000002, 0x10243052, &Beckhoff_EL4132},
    {0x00000002, 0x13893052, &Beckhoff_EL5001},
    {0x00000002, 0x13ED3052, &Beckhoff_EL5101},
    {0x00000002, 0x19C93052, &Beckhoff_EL6601},
    {0x000000D4, 0x00000017, &TR_Electronic_LinEnc2},
    {}
}
```

array with slave descriptions

Definition at line 260 of file types.c.

11.25 types.h File Reference

11.25.1 Detailed Description

EtherCAT slave types.

Definition in file **types.h**.

Data Structures

- struct **ec_field_t**
  Process data field.
- struct **ec_sync_t**
  Sync manager.
- struct **ec_slave_type**
  Slave description type.
- struct **ec_slave_ident_t**
  Slave type identification.

Defines

- #define **EC_MAX_FIELDS** 10
  maximal number of data fields per sync manager
- #define **EC_MAX_SYNC** 16
  maximal number of sync managers per type

Typedefs

- typedef **ec_slave_type ec_slave_type_t**
  Slave description type.

Enumerations

- enum **ec_special_type_t** { **EC_TYPE_NORMAL**, **EC_TYPE_BUS_COUPLER**, **EC_TYPE_-INFRA**, **EC_TYPE_EOE** }
  Special slaves.

Variables

- **ec_slave_ident_t slave_idents** [ ]
  array with slave descriptions

11.25.2 Enumeration Type Documentation

11.25.2.1 enum ec_special_type_t

Special slaves.

Enumerator:

EC_TYPE_NORMAL no special slave

EC_TYPE_BUS_COUPLER slave is a bus coupler

EC_TYPE_INFRA infrastructure slaves, that contain no process data

EC_TYPE_EOE slave is an EoE switch

Definition at line 59 of file types.h.

Chapter 12

IgH EtherCAT master Page Documentation

12.1 Todo List

Global ec_slave_fetch_categories(p. 149) memory allocation

Global ec_slave_locate_string(p. 153) documentation

Global ecrt_slave_sdo_read(p. 17) Make size non-pointer.

Index