IgH EtherCAT master Reference Manual

1.1

Generated by Doxygen 1.4.6

Fri Sep 1 14:56:56 2006

Contents

1 The IgH EtherCAT master **1**

1.1 General information 1

1.2 Contact 1

1.3 License 1

2 IgH EtherCAT master Module Index **3**

2.1 IgH EtherCAT master Modules 3

3 IgH EtherCAT master Directory Hierarchy **5**

3.1 IgH EtherCAT master Directories 5

4 IgH EtherCAT master Hierarchical Index **7**

4.1 IgH EtherCAT master Class Hierarchy 7

5 IgH EtherCAT master Data Structure Index **9**

5.1 IgH EtherCAT master Data Structures 9

6 IgH EtherCAT master File Index **11**

6.1 IgH EtherCAT master File List 11

7 IgH EtherCAT master Page Index **13**

7.1 IgH EtherCAT master Related Pages 13

8 IgH EtherCAT master Module Documentation **15**

8.1 EtherCAT realtime interface 15

8.2 EtherCAT device interface 22

9 IgH EtherCAT master Directory Documentation **25**

9.1 devices/ Directory Reference 25

9.2 include/ Directory Reference 26

9.3 master/ Directory Reference 27

10 IgH EtherCAT master Data Structure Documentation **29**

10.1 ec_address_t Union Reference 29
10.2 ec_code_msg_t Struct Reference 30
10.3 ec_data_reg_t Struct Reference 31
10.4 ec_datagram_t Struct Reference 32
10.5 ec_debug_t Struct Reference 33
10.6 ec_device Struct Reference 34
10.7 ec_domain Struct Reference 35
10.8 ec_eoe Struct Reference 36
10.9 ec_eoe_frame_t Struct Reference 38
10.10 ec_fmmu_t Struct Reference 39
10.11 ec_fsm Struct Reference 40
10.12 ec_master Struct Reference 42
10.13 ec_pdo_reg_t Struct Reference 44
10.14 ec_sdo_data_t Struct Reference 45
10.15 ec_sdo_entry_t Struct Reference 46
10.16 ec_sdo_t Struct Reference 47
10.17 ec_sii_pdo_entry_t Struct Reference 48
10.18 ec_sii_pdo_t Struct Reference 49
10.19 ec_sii_string_t Struct Reference 50
10.20 ec_sii_sync_t Struct Reference 51
10.21 ec_slave Struct Reference 52
10.22 ec_stats_t Struct Reference 56
10.23 ec_varsize_t Struct Reference 57

11 IgH EtherCAT master File Documentation **59**

11.1 datagram.c File Reference 59
11.2 datagram.h File Reference 64
11.3 debug.c File Reference 69
11.4 debug.h File Reference 71
11.5 device.c File Reference 72
11.6 device.h File Reference 75
11.7 domain.c File Reference 78
11.8 domain.h File Reference 82
11.9 ecdb.h File Reference 84
11.10 ecdev.h File Reference 85
11.11 ecrt.h File Reference 86

11.12ethernet.c File Reference 93
11.13ethernet.h File Reference 98
11.14fsm.c File Reference 100
11.15fsm.h File Reference 115
11.16globals.h File Reference 118
11.17mailbox.c File Reference 122
11.18mailbox.h File Reference 124
11.19master.c File Reference 126
11.20master.h File Reference 134
11.21module.c File Reference 138
11.22slave.c File Reference 141
11.23slave.h File Reference 147

12 IgH EtherCAT master Page Documentation **153**

12.1 Todo List 153

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:

EtherCAT realtime interface 15
EtherCAT device interface 22

Chapter 3

IgH EtherCAT master Directory Hierarchy

3.1 IgH EtherCAT master Directories

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

devices 25
include 26
master 27

Chapter 4

IgH EtherCAT master Hierarchical Index

4.1 IgH EtherCAT master Class Hierarchy

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

ec_address_t 29
ec_code_msg_t 30
ec_data_reg_t 31
ec_datagram_t 32
ec_debug_t 33
ec_device 34
ec_domain 35
ec_eoe 36
ec_eoe_frame_t 38
ec_fmmu_t 39
ec_fsm 40
ec_master 42
ec_pdo_reg_t 44
ec_sdo_data_t 45
ec_sdo_entry_t 46
ec_sdo_t 47
ec_sii_pdo_entry_t 48
ec_sii_pdo_t 49
ec_sii_string_t 50
ec_sii_sync_t 51
ec_slave 52
ec_stats_t 56
ec_varsize_t 57

Chapter 5

IgH EtherCAT master Data Structure Index

5.1 IgH EtherCAT master Data Structures

Here are the data structures with brief descriptions:

ec_address_t (EtherCAT address) . . . 29
ec_code_msg_t (Code - Message pair) . . . 30
ec_data_reg_t (Data registration type) . . . 31
ec_datagram_t (EtherCAT datagram) . . . 32
ec_debug_t (Debugging network interface) . . . 33
ec_device (EtherCAT device) . . . 34
ec_domain (EtherCAT domain) . . . 35
ec_eoe (Ethernet-over-EtherCAT (EoE) handler) . . . 36
ec_eoe_frame_t (Queued frame structure) . . . 38
ec_fmmu_t (FMMU configuration) . . . 39
ec_fsm (Finite state machine of an EtherCAT master) . . . 40
ec_master (EtherCAT master) . . . 42
ec_pdo_reg_t (Initialization type for PDO registrations) . . . 44
ec_sdo_data_t . . . 45
ec_sdo_entry_t (CANopen SDO entry) . . . 46
ec_sdo_t (CANopen SDO) . . . 47
ec_sii_pdo_entry_t (PDO entry description (EEPROM)) . . . 48
ec_sii_pdo_t (PDO description (EEPROM)) . . . 49
ec_sii_string_t (String object (EEPROM)) . . . 50
ec_sii_sync_t (Sync manager configuration (EEPROM)) . . . 51
ec_slave (EtherCAT slave) . . . 52
ec_stats_t (Cyclic statistics) . . . 56
ec_varsize_t (Variable-sized field information) . . . 57

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:

datagram.c (Methods of an EtherCAT datagram) . . . 59
datagram.h (EtherCAT datagram structure) . . . 64
debug.c (Ethernet interface for debugging purposes) . . . 69
debug.h (Network interface for debugging purposes) . . . 71
device.c (EtherCAT device methods) . . . 72
device.h (EtherCAT device structure) . . . 75
domain.c (EtherCAT domain methods) . . . 78
domain.h (EtherCAT domain structure) . . . 82
doxygen.c . . . **??**
ecdb.h (EtherCAT Slave Database) . . . 84
ecdev.h (EtherCAT interface for EtherCAT device drivers) . . . 85
ecrt.h (EtherCAT realtime interface) . . . 86
ethernet.c (Ethernet-over-EtherCAT (EoE)) . . . 93
ethernet.h (Ethernet-over-EtherCAT (EoE)) . . . 98
fsm.c (EtherCAT finite state machines) . . . 100
fsm.h (EtherCAT finite state machines) . . . 115
globals.h (Global definitions and macros) . . . 118
mailbox.c (Mailbox functionality) . . . 122
mailbox.h (Mailbox functionality) . . . 124
master.c (EtherCAT master methods) . . . 126
master.h (EtherCAT master structure) . . . 134
module.c (EtherCAT master driver module) . . . 138
slave.c (EtherCAT slave methods) . . . 141
slave.h (EtherCAT stave structure) . . . 147

Chapter 7

IgH EtherCAT master Page Index

7.1 IgH EtherCAT master Related Pages

Here is a list of all related documentation pages:

Todo List 153

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

- **ec_slave_t** ∗ **ecrt_domain_register_pdo** (**ec_domain_t** ∗domain, const char ∗address, uint32_t vendor_id, uint32_t product_code, uint16_t pdo_index, uint8_t pdo_subindex, void ∗∗data_ptr)

  Registers a PDO in a domain.

- int **ecrt_domain_register_pdo_list** (**ec_domain_t** ∗domain, const **ec_pdo_reg_t** ∗pdos)

  Registeres a bunch of data fields.

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

  Processes received process data and requeues the domain datagram(s).

- int **ecrt_domain_state** (const **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.

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

  Asynchronous sending of datagrams.

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

  Asynchronous receiving of datagrams.

- void **ecrt_master_prepare** (**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.

- **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_conf_sdo8** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t value)
- int **ecrt_slave_conf_sdo16** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t value)
- int **ecrt_slave_conf_sdo32** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t value)
- int **ecrt_slave_pdo_size** (**ec_slave_t** ∗slave, uint16_t pdo_index, uint8_t pdo_subindex, size_t size)

8.1.2 Function Documentation

8.1.2.1 ec_slave_t∗ ecrt_domain_register_pdo (ec_domain_t ∗ *domain*, const char ∗ *address*, uint32_t *vendor_id*, uint32_t *product_code*, uint16_t *pdo_index*, uint8_t *pdo_subindex*, void ∗∗ *data_ptr*)

Registers a PDO in a domain.

- If *data_ptr* is NULL, the slave is only validated.

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. 19)

vendor_id vendor ID
product_code product code
pdo_index PDO index
pdo_subindex PDO subindex
data_ptr address of the process data pointer

Definition at line 416 of file domain.c.

8.1.2.2 int ecrt_domain_register_pdo_list (ec_domain_t * *domain*, const ec_pdo_reg_t * *pdos*)

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
pdos array of PDO registrations

Definition at line 485 of file domain.c.

8.1.2.3 void ecrt_domain_process (ec_domain_t * *domain*)

Processes received process data and requeues the domain datagram(s).

Parameters:
domain EtherCAT domain

Definition at line 512 of file domain.c.

8.1.2.4 int ecrt_domain_state (const 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 555 of file domain.c.

8.1.2.5 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 1112 of file master.c.

8.1.2.6 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 1158 of file master.c.

8.1.2.7 void ecrt_master_deactivate (ec_master_t * *master*)

Resets all slaves to INIT state.

Parameters:
master EtherCAT master

Definition at line 1209 of file master.c.

8.1.2.8 void ecrt_master_send (ec_master_t * *master*)

Asynchronous sending of datagrams.

Parameters:
master EtherCAT master

Definition at line 1262 of file master.c.

8.1.2.9 void ecrt_master_receive (ec_master_t * *master*)

Asynchronous receiving of datagrams.

Parameters:
master EtherCAT master

Definition at line 1289 of file master.c.

8.1.2.10 void ecrt_master_prepare (ec_master_t * *master*)

Prepares synchronous IO.

Queues all domain datagrams and sends them. Then waits a certain time, so that **ecrt_master_receive()**(p. 18) can be called securely.

Parameters:
master EtherCAT master

Definition at line 1326 of file master.c.

8.1.2.11 void ecrt_master_run (ec_master_t * *master*)

Does all cyclic master work.

Parameters:
master EtherCAT master

Definition at line 1354 of file master.c.

8.1.2.12 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 1378 of file master.c.

8.1.2.13 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 1480 of file master.c.

8.1.2.14 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 407 of file module.c.

8.1.2.15 void ecrt_release_master (ec_master_t * *master*)

Releases a reserved EtherCAT master.

Parameters:
master EtherCAT master

Definition at line 466 of file module.c.

8.1.2.16 int ecrt_slave_conf_sdo8 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint8_t *value*)

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave
sdo_index SDO index
sdo_subindex SDO subindex
value new SDO value

Definition at line 915 of file slave.c.

8.1.2.17 int ecrt_slave_conf_sdo16 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint16_t *value*)

Returns:
0 in case of success, else < 0

Parameters:

slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

value new SDO value

Definition at line 933 of file slave.c.

8.1.2.18 int ecrt_slave_conf_sdo32 (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, uint32_t *value*)

Returns:

0 in case of success, else < 0

Parameters:

slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

value new SDO value

Definition at line 951 of file slave.c.

8.1.2.19 int ecrt_slave_pdo_size (ec_slave_t * *slave*, uint16_t *pdo_index*, uint8_t *pdo_subindex*, size_t *size*)

Returns:

0 in case of success, else < 0

Parameters:

slave EtherCAT slave

pdo_index PDO index

pdo_subindex PDO subindex

size new PDO size

Definition at line 969 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 262 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 287 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. 24) 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 334 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 362 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 386 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 **ecdb.h**

  EtherCAT Slave Database.

- file **ecrt.h**

  EtherCAT realtime interface.

9.3 master/ Directory Reference

Files

- 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.

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 188 of file globals.h.

Data Fields

- uint32_t **code**

  code

- const char * **message**

  message belonging to code

10.3 ec_data_reg_t Struct Reference

10.3.1 Detailed Description

Data registration type.

Definition at line 53 of file domain.c.

Data Fields

- list_head **list**
  list item
- **ec_slave_t** * **slave**
  slave
- const **ec_sii_sync_t** * **sync**
  sync manager
- off_t **sync_offset**
  pdo offset
- void ** **data_ptr**
  pointer to process data pointer(s)

10.4 ec_datagram_t Struct Reference

10.4.1 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 **cycles_sent**
  time, the datagram was sent

10.5 ec_debug_t Struct Reference

10.5.1 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

10.6 ec_device Struct Reference

10.6.1 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

10.7 ec_domain Struct Reference

10.7.1 Detailed Description

EtherCAT domain.

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

Definition at line 59 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 **data_regs**
  PDO data registrations.
- unsigned int **working_counter_changes**
  working counter changes since last notification
- unsigned long **notify_jiffies**
  time of last notification

10.8 ec_eoe Struct Reference

10.8.1 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
- unsigned long **rate_jiffies**
  time of last rate output
- 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
- uint32_t **rx_counter**
  octets received during last second

- uint32_t **rx_rate**
  receive rate (bps)
- 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
- uint32_t **tx_counter**
  octets transmitted during last second
- uint32_t **tx_rate**
  transmit rate (bps)

10.9 ec_eoe_frame_t Struct Reference

10.9.1 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

10.10 ec_fmmu_t Struct Reference

10.10.1 Detailed Description

FMMU configuration.

Definition at line 218 of file slave.h.

Data Fields

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

10.11 ec_fsm Struct Reference

10.11.1 Detailed Description

Finite state machine of an EtherCAT master.

Definition at line 57 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

- **ec_slave_state_t change_new**
  input: new state
- unsigned long **change_jiffies**
  change timer
- void(∗ **coe_state**)(**ec_fsm_t** ∗)
  CoE state function.
- **ec_sdo_data_t** ∗ **sdodata**
  input/output: SDO data object
- cycles_t **coe_start**
  CoE timestamp.

10.12 ec_master Struct Reference

10.12.1 Detailed Description

EtherCAT master.

Manages slaves, domains and IO.

Definition at line 91 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
- **ec_device_t** * **device**
  EtherCAT device.
- **ec_fsm_t fsm**
  master state machine
- **ec_master_mode_t mode**
  master mode
- list_head **slaves**
  list of slaves on the bus
- unsigned int **slave_count**
  number of slaves on the bus
- list_head **datagram_queue**
  datagram queue
- uint8_t **datagram_index**
  current datagram index
- list_head **domains**
  list of domains
- 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
- uint32_t **idle_cycle_times** [HZ]
  Idle cycle times ring.
- unsigned int **idle_cycle_time_pos**
  time ring buffer position
- timer_list **eoe_timer**
  EoE timer object.
- uint32_t **eoe_cycle_times** [HZ]
  EoE cycle times ring.
- unsigned int **eoe_cycle_time_pos**
  time ring buffer position
- unsigned int **eoe_running**
  non-zero, if EoE processing is active.
- unsigned int **eoe_checked**
  non-zero, if EoE processing is not necessary.
- 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

10.13 ec_pdo_reg_t Struct Reference

10.13.1 Detailed Description

Initialization type for PDO registrations.

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

Definition at line 77 of file ecrt.h.

Data Fields

- const char * **slave_address**
  slave address string (see **ecrt_master_get_slave()**(p. 19)*)*
- uint32_t **vendor_id**
  vendor ID
- uint32_t **product_code**
  product code
- uint16_t **pdo_index**
  PDO index.
- uint8_t **pdo_subindex**
  PDO subindex.
- void ** **data_ptr**
  address of the process data pointer

10.14 ec_sdo_data_t Struct Reference

10.14.1 Detailed Description

Definition at line 202 of file slave.h.

Data Fields

- list_head **list**
  list item
- uint16_t **index**
  SDO index.
- uint8_t **subindex**
  SDO subindex.
- uint8_t * **data**
  pointer to SDO data
- size_t **size**
  size of SDO data

10.15 ec_sdo_entry_t Struct Reference

10.15.1 Detailed Description

CANopen SDO entry.

Definition at line 190 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

10.16 ec_sdo_t Struct Reference

10.16.1 Detailed Description

CANopen SDO.

Definition at line 174 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

10.17 ec_sii_pdo_entry_t Struct Reference

10.17.1 Detailed Description

PDO entry description (EEPROM).

Definition at line 158 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

10.18 ec_sii_pdo_t Struct Reference

10.18.1 Detailed Description

PDO description (EEPROM).

Definition at line 141 of file slave.h.

Data Fields

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

10.19 ec_sii_string_t Struct Reference

10.19.1 Detailed Description

String object (EEPROM).

Definition at line 97 of file slave.h.

Data Fields

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

10.20 ec_sii_sync_t Struct Reference

10.20.1 Detailed Description

Sync manager configuration (EEPROM).

Definition at line 111 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

10.21 ec_slave Struct Reference

10.21.1 Detailed Description

EtherCAT slave.

Definition at line 246 of file slave.h.

Data Fields

- list_head **list**
  list item
- kobject **kobj**
  kobject
- **ec_master_t** ∗ **master**
  master owning the slave
- **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.
- uint8_t **registered**
  true, if slave has been registered
- 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
- uint8_t * **eeprom_data**
  Complete EEPROM image.
- uint16_t **eeprom_size**
  size of the EEPROM contents in byte
- uint16_t * **new_eeprom_data**
  new EEPROM data to write
- uint16_t **new_eeprom_size**
  size of new EEPROM data in words
- 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
- list_head **sii_strings**
  EEPROM STRING categories.
- list_head **sii_syncs**
  EEPROM SYNC MANAGER categories.
- list_head **sii_pdos**
  EEPROM [RT]XPDO categories.
- char ∗ **sii_group**
  slave group acc.
- char ∗ **sii_image**
  slave image name acc.
- char ∗ **sii_order**
  slave order number acc.
- char ∗ **sii_name**
  slave name acc.
- **ec_fmmu_t fmmus** [EC_MAX_FMMUS]
  FMMU configurations.
- uint8_t **fmmu_count**
  number of FMMUs used
- list_head **sdo_dictionary**
  SDO directory list.
- list_head **sdo_confs**
  list of SDO configurations
- list_head **varsize_fields**
  size information for variable-sized data fields.

10.21.2 Field Documentation

10.21.2.1 char* ec_slave::sii_group

slave group acc.

to EEPROM

Definition at line 297 of file slave.h.

10.21.2.2 char* ec_slave::sii_image

slave image name acc.

to EEPROM

Definition at line 298 of file slave.h.

10.21.2.3 char* ec_slave::sii_order

slave order number acc.

to EEPROM

Definition at line 299 of file slave.h.

10.21.2.4 char* ec_slave::sii_name

slave name acc.

to EEPROM

Definition at line 300 of file slave.h.

10.22 ec_stats_t Struct Reference

10.22.1 Detailed Description

Cyclic statistics.

Definition at line 72 of file master.h.

Data Fields

- unsigned int **timeouts**
  datagram timeouts
- unsigned int **corrupted**
  corrupted frames
- unsigned int **skipped**
  skipped datagrams (the ones that were requeued when not yet received)
- unsigned int **unmatched**
  unmatched datagrams (received, but not queued any longer)
- unsigned long **output_jiffies**
  time of last output

10.23 ec_varsize_t Struct Reference

10.23.1 Detailed Description

Variable-sized field information.

Definition at line 232 of file slave.h.

Data Fields

- list_head **list**
  list item
- const **ec_sii_pdo_t** * **pdo**
  PDO.
- size_t **size**
  field size

Chapter 11

IgH EtherCAT master File Documentation

11.1 datagram.c File Reference

11.1.1 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.

11.1.2 Function Documentation

11.1.2.1 void ec_datagram_init (ec_datagram_t * *datagram*)

Datagram constructor.

Parameters:
datagram EtherCAT datagram

Definition at line 70 of file datagram.c.

11.1.2.2 void ec_datagram_clear (ec_datagram_t * *datagram*)

Datagram destructor.

Parameters:
datagram EtherCAT datagram

Definition at line 89 of file datagram.c.

11.1.2.3 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.

11.1.2.4 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.

11.1.2.5 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.

11.1.2.6 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.

11.1.2.7 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.

11.1.2.8 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.

11.1.2.9 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.

11.1.2.10 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.

11.2 datagram.h File Reference

11.2.1 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_DATAGRAM_NONE = 0x00, **EC_DATAGRAM_APRD** = 0x01, **EC_DATAGRAM_APWR** = 0x02, **EC_DATAGRAM_NPRD** = 0x04,

  EC_DATAGRAM_NPWR = 0x05, **EC_DATAGRAM_BRD** = 0x07, **EC_DATAGRAM_BWR** = 0x08, **EC_DATAGRAM_LRW** = 0x0C }

  EtherCAT datagram type.

- enum **ec_datagram_state_t** {

  EC_DATAGRAM_INIT, **EC_DATAGRAM_QUEUED**, **EC_DATAGRAM_SENT**, **EC_DATAGRAM_RECEIVED**,

  EC_DATAGRAM_TIMED_OUT, **EC_DATAGRAM_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.

11.2.2 Enumeration Type Documentation

11.2.2.1 enum ec_datagram_type_t

EtherCAT datagram type.

Enumerator:

EC_DATAGRAM_NONE Dummy.
EC_DATAGRAM_APRD Auto-increment physical read.
EC_DATAGRAM_APWR Auto-increment physical write.
EC_DATAGRAM_NPRD Node-addressed physical read.
EC_DATAGRAM_NPWR Node-addressed physical write.
EC_DATAGRAM_BRD Broadcast read.
EC_DATAGRAM_BWR Broadcast write.
EC_DATAGRAM_LRW Logical read/write.

Definition at line 56 of file datagram.h.

11.2.2.2 enum ec_datagram_state_t

EtherCAT datagram state.

Enumerator:

EC_DATAGRAM_INIT new datagram
EC_DATAGRAM_QUEUED datagram queued by master
EC_DATAGRAM_SENT datagram has been sent and still in the queue
EC_DATAGRAM_RECEIVED datagram has been received and dequeued
EC_DATAGRAM_TIMED_OUT datagram timed out and was dequeued
EC_DATAGRAM_ERROR error while sending/receiving, datagram dequeued

Definition at line 73 of file datagram.h.

11.2.3 Function Documentation

11.2.3.1 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.

11.2.3.2 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.

11.2.3.3 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.

11.2.3.4 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.

11.2.3.5 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.

11.2.3.6 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.

11.2.3.7 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.

11.2.3.8 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.

11.3 debug.c File Reference

11.3.1 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.

11.3.2 Function Documentation

11.3.2.1 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.

11.3.2.2 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.

11.3.2.3 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.

11.3.2.4 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.

11.3.2.5 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.

11.3.2.6 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.

11.4 debug.h File Reference

11.4.1 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.

11.4.2 Function Documentation

11.4.2.1 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.

11.4.2.2 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.

11.5 device.c File Reference

11.5.1 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.

11.5.2 Function Documentation

11.5.2.1 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.

11.5.2.2 void ec_device_clear (ec_device_t * *device*)

EtherCAT device destuctor.

Parameters:
device EtherCAT device

Definition at line 107 of file device.c.

11.5.2.3 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.

11.5.2.4 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.

11.5.2.5 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.

11.5.2.6 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.

11.5.2.7 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.

11.6 device.h File Reference

11.6.1 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.

11.6.2 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 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.

11.6.2.3 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.

11.6.2.4 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.

11.6.2.5 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.

11.7 domain.c File Reference

11.7.1 Detailed Description

EtherCAT domain methods.

Definition in file **domain.c**.

Data Structures

- struct **ec_data_reg_t**

  Data registration type.

Functions

- void **ec_domain_clear_data_regs** (**ec_domain_t** ∗domain)

  Clears the list of the data registrations.

- 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_pdo_entry** (**ec_domain_t** ∗domain, **ec_slave_t** ∗slave, const **ec_sii_pdo_t** ∗pdo, const **ec_sii_pdo_entry_t** ∗entry, void ∗∗data_ptr)

  Registeres a PDO entry.

- 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_queue** (**ec_domain_t** ∗domain)

  Places all process data datagrams in the masters datagram queue.

- **ec_slave_t** ∗ **ecrt_domain_register_pdo** (**ec_domain_t** ∗domain, const char ∗address, uint32_t vendor_id, uint32_t product_code, uint16_t pdo_index, uint8_t pdo_subindex, void ∗∗data_ptr)

  Registers a PDO in a domain.

- int **ecrt_domain_register_pdo_list** (**ec_domain_t** ∗domain, const **ec_pdo_reg_t** ∗pdos)

  Registeres a bunch of data fields.

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

Processes received process data and requeues the domain datagram(s).

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

  Returns the state of a domain.

11.7.2 Function Documentation

11.7.2.1 void ec_domain_clear_data_regs (ec_domain_t *)

Clears the list of the data registrations.

Parameters:
domain EtherCAT domain

Definition at line 233 of file domain.c.

11.7.2.2 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 391 of file domain.c.

11.7.2.3 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 99 of file domain.c.

11.7.2.4 void ec_domain_clear (struct kobject * *kobj*)

Domain destructor.

Parameters:
kobj kobject of the domain

Definition at line 134 of file domain.c.

11.7.2.5 int ec_domain_reg_pdo_entry (ec_domain_t * *domain*, ec_slave_t * *slave*, const ec_sii_pdo_t * *pdo*, const ec_sii_pdo_entry_t * *entry*, void ** *data_ptr*)

Registeres a PDO entry.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain
slave slave
pdo PDO
entry PDO registration entry
data_ptr pointer to the process data pointer

Definition at line 160 of file domain.c.

11.7.2.6 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 250 of file domain.c.

11.7.2.7 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 process data.

Returns:
0 in case of success, else < 0

Parameters:

domain EtherCAT domain

base_address logical base address

Definition at line 283 of file domain.c.

11.7.2.8 void ec_domain_queue (ec_domain_t * *domain*)

Places all process data datagrams in the masters datagram queue.

Parameters:

domain EtherCAT domain

Definition at line 375 of file domain.c.

11.8 domain.h File Reference

11.8.1 Detailed Description

EtherCAT domain structure.

Definition in file **domain.h**.

Data Structures

- 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.

- void **ec_domain_queue** (**ec_domain_t** ∗)

  Places all process data datagrams in the masters datagram queue.

11.8.2 Function Documentation

11.8.2.1 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 99 of file domain.c.

11.8.2.2 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 process data.

Returns:
0 in case of success, else < 0

Parameters:
domain EtherCAT domain

base_address logical base address

Definition at line 283 of file domain.c.

11.9 ecdb.h File Reference

11.9.1 Detailed Description

EtherCAT Slave Database.

Definition in file **ecdb.h**.

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. 34)

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_pdo_reg_t**
  Initialization type for PDO 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.
- void **ecrt_master_prepare** (**ec_master_t** ∗master)
  Prepares synchronous IO.
- void **ecrt_master_send** (**ec_master_t** ∗master)
  Asynchronous sending of datagrams.
- void **ecrt_master_receive** (**ec_master_t** ∗master)
  Asynchronous receiving of datagrams.
- void **ecrt_master_run** (**ec_master_t** ∗master)
  Does all cyclic master work.

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

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

- int **ecrt_master_state** (const **ec_master_t** ∗master)
- **ec_slave_t** ∗ **ecrt_domain_register_pdo** (**ec_domain_t** ∗domain, const char ∗address, uint32_t vendor_id, uint32_t product_code, uint16_t pdo_index, uint8_t pdo_subindex, void ∗∗data_ptr)

  Registers a PDO in a domain.

- int **ecrt_domain_register_pdo_list** (**ec_domain_t** ∗domain, const **ec_pdo_reg_t** ∗pdos)

  Registeres a bunch of data fields.

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

  Processes received process data and requeues the domain datagram(s).

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

  Returns the state of a domain.

- int **ecrt_slave_conf_sdo8** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t value)
- int **ecrt_slave_conf_sdo16** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t value)
- int **ecrt_slave_conf_sdo32** (**ec_slave_t** ∗slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t value)
- int **ecrt_slave_pdo_size** (**ec_slave_t** ∗slave, uint16_t pdo_index, uint8_t pdo_subindex, 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 160 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 169 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 184 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 193 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 202 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 211 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 220 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 229 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 243 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 254 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 262 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 274 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 282 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 294 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. 42)

Definition at line 63 of file ecrt.h.

11.11.3.2 typedef struct ec_domain ec_domain_t

See also:
ec_domain(p. 35)

Definition at line 66 of file ecrt.h.

11.11.3.3 typedef struct ec_slave ec_slave_t

See also:
ec_slave(p. 52)

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.
- int **ec_eoe_active** (const **ec_eoe_t** ∗eoe)
  Returns the state of the device.

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 186 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 318 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 336 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 362 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 509 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 572 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 615 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 638 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 661 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 711 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 162 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 208 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 279 of file ethernet.c.

11.12.3.15 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 303 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.

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

  Returns the state of the device.

11.13.2 Typedef Documentation

11.13.2.1 typedef struct ec_eoe ec_eoe_t

See also:
ec_eoe(p. 36)

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 162 of file ethernet.c.

11.13.3.3 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 303 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_startup_start** (**ec_fsm_t** ∗fsm)
  Master state: START.
- void **ec_fsm_startup_broadcast** (**ec_fsm_t** ∗fsm)
  Master state: BROADCAST.
- void **ec_fsm_startup_scan** (**ec_fsm_t** ∗fsm)
  Master state: SCAN.
- void **ec_fsm_configuration_start** (**ec_fsm_t** ∗fsm)
  Master configuration state machine: START.
- void **ec_fsm_configuration_conf** (**ec_fsm_t** ∗fsm)

Master state: CONF.

- 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_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_sdoconf** (**ec_fsm_t** ∗fsm)
  Slave state: SDOCONF.
- 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_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_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_coe_down_start** (**ec_fsm_t** *fsm)
  CoE state: DOWN_START.
- void **ec_fsm_coe_down_request** (**ec_fsm_t** *fsm)
  CoE state: DOWN_REQUEST.
- void **ec_fsm_coe_down_check** (**ec_fsm_t** *fsm)
  CoE state: DOWN_CHECK.
- void **ec_fsm_coe_down_response** (**ec_fsm_t** *fsm)
  CoE state: DOWN_RESPONSE.
- void **ec_fsm_end** (**ec_fsm_t** *fsm)
  State: END.
- void **ec_fsm_error** (**ec_fsm_t** *fsm)
  State: ERROR.
- void **ec_canopen_abort_msg** (uint32_t abort_code)
  Outputs an SDO abort message.

- 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_startup** (**ec_fsm_t** ∗fsm)
  Initializes the master startup state machine.
- int **ec_fsm_startup_running** (**ec_fsm_t** ∗fsm)
  Returns the running state of the master startup state machine.
- int **ec_fsm_startup_success** (**ec_fsm_t** ∗fsm)
  Returns, if the master startup state machine terminated with success.
- void **ec_fsm_configuration** (**ec_fsm_t** ∗fsm)
  Initializes the master configuration state machine.
- int **ec_fsm_configuration_running** (**ec_fsm_t** ∗fsm)
  Returns the running state of the master configuration state machine.
- int **ec_fsm_configuration_success** (**ec_fsm_t** ∗fsm)
  Returns, if the master confuguration state machine terminated with success.
- 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.
- const **ec_code_msg_t sdo_abort_messages** [ ]
  SDO abort 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 422 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 436 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 668 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 719 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 789 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 839 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 918 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 871 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 936 of file fsm.c.

11.14.2.10 void ec_fsm_startup_start (ec_fsm_t * *fsm*)

Master state: START.

Starts with getting slave count and slave states.

Definition at line 246 of file fsm.c.

11.14.2.11 void ec_fsm_startup_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 260 of file fsm.c.

11.14.2.12 void ec_fsm_startup_scan (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 324 of file fsm.c.

11.14.2.13 void ec_fsm_configuration_start (ec_fsm_t *)

Master configuration state machine: START.

Parameters:
fsm finite state machine

Definition at line 362 of file fsm.c.

11.14.2.14 void ec_fsm_configuration_conf (ec_fsm_t *)

Master state: CONF.

Parameters:
fsm finite state machine

Definition at line 386 of file fsm.c.

11.14.2.15 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 986 of file fsm.c.

11.14.2.16 void ec_fsm_slavescan_address (ec_fsm_t *)

Slave state: ADDRESS.

Parameters:
fsm finite state machine

Definition at line 1003 of file fsm.c.

11.14.2.17 void ec_fsm_slavescan_state (ec_fsm_t *)

Slave state: STATE.

Parameters:
fsm finite state machine

Definition at line 1028 of file fsm.c.

11.14.2.18 void ec_fsm_slavescan_base (ec_fsm_t *)

Slave state: BASE.

Parameters:
fsm finite state machine

Definition at line 1061 of file fsm.c.

11.14.2.19 void ec_fsm_slavescan_datalink (ec_fsm_t *)

Slave state: DATALINK.

Parameters:
fsm finite state machine

Definition at line 1096 of file fsm.c.

11.14.2.20 void ec_fsm_slavescan_eeprom_size (ec_fsm_t *)

Slave state: EEPROM_SIZE.

Parameters:
fsm finite state machine

Definition at line 1134 of file fsm.c.

11.14.2.21 void ec_fsm_slavescan_eeprom_data (ec_fsm_t *)

Slave state: EEPROM_DATA.

Parameters:
fsm finite state machine

Definition at line 1194 of file fsm.c.

11.14.2.22 void ec_fsm_slaveconf_init (ec_fsm_t *)

Slave state: INIT.

Parameters:
fsm finite state machine

Definition at line 1308 of file fsm.c.

11.14.2.23 void ec_fsm_slaveconf_sync (ec_fsm_t *)

Slave state: SYNC.

Parameters:
fsm finite state machine

Definition at line 1368 of file fsm.c.

11.14.2.24 void ec_fsm_slaveconf_preop (ec_fsm_t *)

Slave state: PREOP.

Parameters:
fsm finite state machine

Definition at line 1394 of file fsm.c.

11.14.2.25 void ec_fsm_slaveconf_fmmu (ec_fsm_t *)

Slave state: FMMU.

Parameters:
fsm finite state machine

Definition at line 1451 of file fsm.c.

11.14.2.26 void ec_fsm_slaveconf_sdoconf (ec_fsm_t *)

Slave state: SDOCONF.

Parameters:
fsm finite state machine

Definition at line 1487 of file fsm.c.

11.14.2.27 void ec_fsm_slaveconf_saveop (ec_fsm_t *)

Slave state: SAVEOP.

Parameters:
fsm finite state machine

Definition at line 1523 of file fsm.c.

11.14.2.28 void ec_fsm_slaveconf_op (ec_fsm_t *)

Slave state: OP.

Parameters:
fsm finite state machine

Definition at line 1554 of file fsm.c.

11.14.2.29 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 1579 of file fsm.c.

11.14.2.30 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 1605 of file fsm.c.

11.14.2.31 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 1637 of file fsm.c.

11.14.2.32 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 1694 of file fsm.c.

11.14.2.33 void ec_fsm_sii_write_check (ec_fsm_t *)

SII state: WRITE_CHECK.

Parameters:
fsm finite state machine

Definition at line 1714 of file fsm.c.

11.14.2.34 void ec_fsm_sii_write_check2 (ec_fsm_t *)

SII state: WRITE_CHECK2.

Parameters:
fsm finite state machine

Definition at line 1739 of file fsm.c.

11.14.2.35 void ec_fsm_change_start (ec_fsm_t *)

Change state: START.

Parameters:
fsm finite state machine

Definition at line 1777 of file fsm.c.

11.14.2.36 void ec_fsm_change_check (ec_fsm_t *)

Change state: CHECK.

Parameters:
fsm finite state machine

Definition at line 1797 of file fsm.c.

11.14.2.37 void ec_fsm_change_status (ec_fsm_t *)

Change state: STATUS.

Parameters:
fsm finite state machine

Definition at line 1838 of file fsm.c.

11.14.2.38 void ec_fsm_change_code (ec_fsm_t *)

Change state: CODE.

Parameters:
fsm finite state machine

Definition at line 1931 of file fsm.c.

11.14.2.39 void ec_fsm_change_ack (ec_fsm_t *)

Change state: ACK.

Parameters:
fsm finite state machine

Definition at line 1969 of file fsm.c.

11.14.2.40 void ec_fsm_change_check_ack (ec_fsm_t *)

Change state: CHECK ACK.

Parameters:
fsm finite state machine

Definition at line 1995 of file fsm.c.

11.14.2.41 void ec_fsm_coe_down_start (ec_fsm_t *)

CoE state: DOWN_START.

Parameters:
fsm finite state machine

Definition at line 2039 of file fsm.c.

11.14.2.42 void ec_fsm_coe_down_request (ec_fsm_t *)

CoE state: DOWN_REQUEST.

Parameters:
fsm finite state machine

Definition at line 2079 of file fsm.c.

11.14.2.43 void ec_fsm_coe_down_check (ec_fsm_t *)

CoE state: DOWN_CHECK.

Parameters:
fsm finite state machine

Definition at line 2104 of file fsm.c.

11.14.2.44 void ec_fsm_coe_down_response (ec_fsm_t *)

CoE state: DOWN_RESPONSE.

Parameters:
fsm finite state machine

Definition at line 2141 of file fsm.c.

11.14.2.45 void ec_fsm_end (ec_fsm_t *)

State: END.

Parameters:
fsm finite state machine

Definition at line 2284 of file fsm.c.

11.14.2.46 void ec_fsm_error (ec_fsm_t *)

State: ERROR.

Parameters:
fsm finite state machine

Definition at line 2274 of file fsm.c.

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

Constructor.

Parameters:
fsm finite state machine

master EtherCAT master

Definition at line 111 of file fsm.c.

11.14.2.48 void ec_fsm_clear (ec_fsm_t * *fsm*)

Destructor.

Parameters:
fsm finite state machine

Definition at line 136 of file fsm.c.

11.14.2.49 void ec_fsm_reset (ec_fsm_t * *fsm*)

Resets the state machine.

Parameters:
fsm finite state machine

Definition at line 147 of file fsm.c.

11.14.2.50 void ec_fsm_execute (ec_fsm_t * *fsm*)

Executes the current state of the state machine.

Parameters:
fsm finite state machine

Definition at line 160 of file fsm.c.

11.14.2.51 int ec_fsm_startup_running (ec_fsm_t * *fsm*)

Returns the running state of the master startup state machine.

Returns:
non-zero if not terminated yet.

Parameters:
fsm Finite state machine

Definition at line 183 of file fsm.c.

11.14.2.52 int ec_fsm_startup_success (ec_fsm_t * *fsm*)

Returns, if the master startup state machine terminated with success.

Returns:
non-zero if successful.

Parameters:
fsm Finite state machine

Definition at line 196 of file fsm.c.

11.14.2.53 int ec_fsm_configuration_running (ec_fsm_t * *fsm*)

Returns the running state of the master configuration state machine.

Returns:
non-zero if not terminated yet.

Parameters:
fsm Finite state machine

Definition at line 219 of file fsm.c.

11.14.2.54 int ec_fsm_configuration_success (ec_fsm_t * *fsm*)

Returns, if the master confuguration state machine terminated with success.

Returns:
non-zero if successful.

Parameters:
fsm Finite state machine

Definition at line 232 of file fsm.c.

11.14.2.55 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 552 of file fsm.c.

11.14.2.56 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 621 of file fsm.c.

11.14.2.57 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 759 of file fsm.c.

11.14.3 Variable Documentation

11.14.3.1 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 2208 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.
- void **ec_fsm_startup** (**ec_fsm_t** ∗)
  Initializes the master startup state machine.
- int **ec_fsm_startup_running** (**ec_fsm_t** ∗)
  Returns the running state of the master startup state machine.
- int **ec_fsm_startup_success** (**ec_fsm_t** ∗)
  Returns, if the master startup state machine terminated with success.
- void **ec_fsm_configuration** (**ec_fsm_t** ∗)
  Initializes the master configuration state machine.
- int **ec_fsm_configuration_running** (**ec_fsm_t** ∗)
  Returns the running state of the master configuration state machine.
- int **ec_fsm_configuration_success** (**ec_fsm_t** ∗)
  Returns, if the master confuguration state machine terminated with success.

11.15.2 Typedef Documentation

11.15.2.1 typedef struct ec_fsm ec_fsm_t

See also:
ec_fsm(p. 40)

Definition at line 51 of file fsm.h.

11.15.3 Function Documentation

11.15.3.1 int ec_fsm_startup_running (ec_fsm_t * *fsm*)

Returns the running state of the master startup state machine.

Returns:
non-zero if not terminated yet.

Parameters:
fsm Finite state machine

Definition at line 183 of file fsm.c.

11.15.3.2 int ec_fsm_startup_success (ec_fsm_t * *fsm*)

Returns, if the master startup state machine terminated with success.

Returns:
non-zero if successful.

Parameters:
fsm Finite state machine

Definition at line 196 of file fsm.c.

11.15.3.3 int ec_fsm_configuration_running (ec_fsm_t * *fsm*)

Returns the running state of the master configuration state machine.

Returns:
non-zero if not terminated yet.

Parameters:
fsm Finite state machine

Definition at line 219 of file fsm.c.

11.15.3.4 int ec_fsm_configuration_success (ec_fsm_t * *fsm*)

Returns, if the master confuguration state machine terminated with success.

Returns:
non-zero if successful.

Parameters:
fsm Finite state machine

Definition at line 232 of file fsm.c.

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** 1
  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_IO_TIMEOUT** 500
  datagram timeout in microseconds
- #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. 121)*, 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.
- size_t **ec_state_string** (uint8_t, char *)
  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 91 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 103 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 113 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 123 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 133 of file globals.h.

11.16.2.6 #define EC_LIT(X) #X

Helper macro for **EC_STR()**(p. 121), literates a macro argument.

Parameters:
X argument to literate.

Definition at line 141 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 148 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 157 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 169 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.

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.

11.18 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.

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.

11.19 master.c File Reference

11.19.1 Detailed Description

EtherCAT master methods.

Definition in file **master.c**.

Functions

- void **ec_master_sync_io** (**ec_master_t** ∗master)
  Sends queued datagrams and waits for their reception.
- 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_datagrams** (**ec_master_t** ∗master, const uint8_t ∗frame_data, size_t size)
  Processes a received frame.
- 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_sii_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.
- ssize_t **ec_master_info** (**ec_master_t** ∗master, char ∗buffer)
  Formats master information for SysFS read access.
- void **ec_master_eoe_start** (**ec_master_t** ∗master)
  Starts Ethernet-over-EtherCAT processing on demand.
- void **ec_master_eoe_stop** (**ec_master_t** ∗master)
  Stops the Ethernet-over-EtherCAT processing.
- void **ec_master_calc_addressing** (**ec_master_t** ∗master)
  Calculates Advanced Position Adresses.
- void **ec_master_measure_bus_time** (**ec_master_t** ∗master)
  Measures the time, a frame is on the bus.
- **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.
- void **ecrt_master_send** (**ec_master_t** ∗master)
  Asynchronous sending of datagrams.
- void **ecrt_master_receive** (**ec_master_t** ∗master)
  Asynchronous receiving of datagrams.
- void **ecrt_master_prepare** (**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.

11.19.2 Function Documentation

11.19.2.1 void ec_master_sync_io (ec_master_t *)

Sends queued datagrams and waits for their reception.

Parameters:
master EtherCAT master

Definition at line 1233 of file master.c.

11.19.2.2 void ec_master_idle_run (void *)

Idle mode function.

Parameters:
data master pointer

Definition at line 605 of file master.c.

11.19.2.3 void ec_master_eoe_run (unsigned *long*)

Does the Ethernet-over-EtherCAT processing.

Parameters:
data master pointer

Definition at line 951 of file master.c.

11.19.2.4 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 772 of file master.c.

11.19.2.5 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 799 of file master.c.

11.19.2.6 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 99 of file master.c.

11.19.2.7 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 184 of file master.c.

11.19.2.8 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 218 of file master.c.

11.19.2.9 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 285 of file master.c.

11.19.2.10 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 312 of file master.c.

11.19.2.11 void ec_master_receive_datagrams (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 frame data

size size of the received data

Definition at line 414 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 501 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 529 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 559 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 581 of file master.c.

11.19.2.16 void ec_sync_config (const ec_sii_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 641 of file master.c.

11.19.2.17 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 672 of file master.c.

11.19.2.18 ssize_t ec_master_info (ec_master_t * *master*, char * *buffer*)

Formats master information for SysFS read access.

Returns:

number of bytes written

Parameters:

master EtherCAT master

buffer memory to store data

Definition at line 700 of file master.c.

11.19.2.19 void ec_master_eoe_start (ec_master_t * *master*)

Starts Ethernet-over-EtherCAT processing on demand.

Parameters:

master EtherCAT master

Definition at line 854 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 922 of file master.c.

11.19.2.21 void ec_master_calc_addressing (ec_master_t * *master*)

Calculates Advanced Position Adresses.

Parameters:
master EtherCAT master

Definition at line 1013 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_OPERATION** }
  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 on demand.
- void **ec_master_eoe_stop** (**ec_master_t** *)
  Stops the Ethernet-over-EtherCAT processing.
- void **ec_master_receive_datagrams** (**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_bus_scan** (**ec_master_t** ∗)
  Scans the EtherCAT bus for slaves.
- void **ec_master_output_stats** (**ec_master_t** ∗)
  Output statistics in cyclic mode.
- void **ec_master_clear_slaves** (**ec_master_t** ∗)
  Clears all slaves.
- void **ec_master_measure_bus_time** (**ec_master_t** ∗)
  Measures the time, a frame is on the bus.
- void **ec_sync_config** (const **ec_sii_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_calc_addressing** (**ec_master_t** ∗)
  Calculates Advanced Position Adresses.

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 99 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 184 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 218 of file master.c.

11.20.2.4 void ec_master_receive_datagrams (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 frame data
size size of the received data

Definition at line 414 of file master.c.

11.20.2.5 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 501 of file master.c.

11.20.2.6 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 529 of file master.c.

11.20.2.7 void ec_sync_config (const ec_sii_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 641 of file master.c.

11.20.2.8 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 672 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.
- size_t **ec_state_string** (uint8_t states, char *buffer)
  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 size_t ec_state_string (uint8_t *states*, char * *buffer*)

Prints slave states in clear text.

Parameters:
states slave states
buffer target buffer (min. 25 bytes)

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

- 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_strings** (**ec_slave_t** *slave, const uint8_t *data)

  Fetches data from a STRING category.

- void **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_sii_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.

- int **ec_slave_prepare_fmmu** (**ec_slave_t** *slave, const **ec_domain_t** *domain, const **ec_sii_sync_t** *sync)

  Prepares an FMMU configuration.

- size_t **ec_slave_info** (const **ec_slave_t** *slave, char *buffer)

  Outputs all information about a certain slave.

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

  Schedules an EEPROM write operation.

- uint16_t **ec_slave_calc_sync_size** (const **ec_slave_t** *slave, const **ec_sii_sync_t** *sync)
  Calculates the size of a sync manager by evaluating PDO sizes.

- int **ec_slave_is_coupler** (const **ec_slave_t** *slave)
- int **ec_slave_conf_sdo** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, const uint8_t *data, size_t size)
- int **ecrt_slave_conf_sdo8** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint8_t value)
- int **ecrt_slave_conf_sdo16** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint16_t value)
- int **ecrt_slave_conf_sdo32** (**ec_slave_t** *slave, uint16_t sdo_index, uint8_t sdo_subindex, uint32_t value)
- int **ecrt_slave_pdo_size** (**ec_slave_t** *slave, uint16_t pdo_index, uint8_t pdo_subindex, 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 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 733 of file slave.c.

11.22.2.2 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 781 of file slave.c.

11.22.2.3 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 94 of file slave.c.

11.22.2.4 void ec_slave_clear (struct kobject * *kobj*)

Slave destructor.

Parameters:
kobj kobject of the slave

Definition at line 177 of file slave.c.

11.22.2.5 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 260 of file slave.c.

11.22.2.6 void 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 298 of file slave.c.

11.22.2.7 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 321 of file slave.c.

11.22.2.8 int ec_slave_fetch_pdo (ec_slave_t * *slave*, const uint8_t * *data*, size_t *word_count*, ec_sii_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 357 of file slave.c.

11.22.2.9 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 419 of file slave.c.

11.22.2.10 int ec_slave_prepare_fmmu (ec_slave_t * *slave*, const ec_domain_t * *domain*, const ec_sii_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. 18). The FMMU configuration is done in a way, that the complete data range of the corresponding sync manager is covered. Seperate FMMUs are 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 475 of file slave.c.

11.22.2.11 size_t ec_slave_info (const ec_slave_t * *slave*, char * *buffer*)

Outputs all information about a certain slave.

Parameters:
slave EtherCAT slave
buffer Output buffer

Definition at line 509 of file slave.c.

11.22.2.12 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 656 of file slave.c.

11.22.2.13 uint16_t ec_slave_calc_sync_size (const ec_slave_t * *slave*, const ec_sii_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 825 of file slave.c.

11.22.2.14 int ec_slave_is_coupler (const ec_slave_t * *slave*)

Returns:
non-zero if slave is a bus coupler

Parameters:
slave EtherCAT slave

Definition at line 858 of file slave.c.

11.22.2.15 int ec_slave_conf_sdo (ec_slave_t * *slave*, uint16_t *sdo_index*, uint8_t *sdo_subindex*, const uint8_t * *data*, size_t *size*)

Returns:
0 in case of success, else < 0

Parameters:
slave EtherCAT slave

sdo_index SDO index

sdo_subindex SDO subindex

data SDO data

size SDO size in bytes

Definition at line 871 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_sii_string_t**

  String object (EEPROM).

- struct **ec_sii_sync_t**

  Sync manager configuration (EEPROM).

- struct **ec_sii_pdo_t**

  PDO description (EEPROM).

- struct **ec_sii_pdo_entry_t**

  PDO entry description (EEPROM).

- struct **ec_sdo_t**

  CANopen SDO.

- struct **ec_sdo_entry_t**

  CANopen SDO entry.

- struct **ec_sdo_data_t**
- struct **ec_fmmu_t**

  FMMU configuration.

- 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_sii_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_prepare_fmmu** (**ec_slave_t** ∗, const **ec_domain_t** ∗, const **ec_sii_sync_t** ∗)
  Prepares an FMMU configuration.

- int **ec_slave_fetch_strings** (**ec_slave_t** ∗, const uint8_t ∗)
  Fetches data from a STRING category.

- void **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_sii_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.

- uint16_t **ec_slave_calc_sync_size** (const **ec_slave_t** ∗, const **ec_sii_sync_t** ∗)
  Calculates the size of a sync manager by evaluating PDO sizes.

- int **ec_slave_is_coupler** (const **ec_slave_t** ∗)

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 58 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 81 of file slave.h.

11.23.2.3 enum ec_sii_pdo_type_t

PDO type.

Enumerator:
EC_RX_PDO Reveive PDO.
EC_TX_PDO Transmit PDO.

Definition at line 128 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 94 of file slave.c.

11.23.3.2 int ec_slave_prepare_fmmu (ec_slave_t * *slave*, const ec_domain_t * *domain*, const ec_sii_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. 18). The FMMU configuration is done in a way, that the complete data range of the corresponding sync manager is covered. Seperate FMMUs are 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 475 of file slave.c.

11.23.3.3 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 260 of file slave.c.

11.23.3.4 void 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 298 of file slave.c.

11.23.3.5 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 321 of file slave.c.

11.23.3.6 int ec_slave_fetch_pdo (ec_slave_t * *slave*, const uint8_t * *data*, size_t *word_count*, ec_sii_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 357 of file slave.c.

11.23.3.7 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 419 of file slave.c.

11.23.3.8 uint16_t ec_slave_calc_sync_size (const ec_slave_t * *slave*, const ec_sii_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 825 of file slave.c.

11.23.3.9 int ec_slave_is_coupler (const ec_slave_t * *slave*)

Returns:
non-zero if slave is a bus coupler

Parameters:
slave EtherCAT slave

Definition at line 858 of file slave.c.

Chapter 12

IgH EtherCAT master Page Documentation

12.1 Todo List

Global ec_slave_locate_string(p. 144) documentation

Index

COMPILE_INFO
    module.c, 139

datagram.c, 59
    ec_datagram_aprd, 61
    ec_datagram_apwr, 61
    ec_datagram_brd, 62
    ec_datagram_bwr, 62
    ec_datagram_clear, 60
    ec_datagram_init, 60
    ec_datagram_lrw, 62
    ec_datagram_nprd, 60
    ec_datagram_npwr, 61
    ec_datagram_prealloc, 60
datagram.h, 64
    EC_DATAGRAM_APRD, 65
    ec_datagram_aprd, 66
    EC_DATAGRAM_APWR, 65
    ec_datagram_apwr, 67
    EC_DATAGRAM_BRD, 65
    ec_datagram_brd, 67
    EC_DATAGRAM_BWR, 65
    ec_datagram_bwr, 67
    EC_DATAGRAM_ERROR, 65
    EC_DATAGRAM_INIT, 65
    EC_DATAGRAM_LRW, 65
    ec_datagram_lrw, 68
    EC_DATAGRAM_NONE, 65
    EC_DATAGRAM_NPRD, 65
    ec_datagram_nprd, 66
    EC_DATAGRAM_NPWR, 65
    ec_datagram_npwr, 66
    ec_datagram_prealloc, 66
    EC_DATAGRAM_QUEUED, 65
    EC_DATAGRAM_RECEIVED, 65
    EC_DATAGRAM_SENT, 65
    ec_datagram_state_t, 65
    EC_DATAGRAM_TIMED_OUT, 65
    ec_datagram_type_t, 65
debug.c, 69
    ec_dbgdev_open, 69
    ec_dbgdev_stats, 69
    ec_dbgdev_stop, 69
    ec_debug_clear, 70
    ec_debug_init, 70
    ec_debug_send, 70
debug.h, 71
    ec_debug_clear, 71
    ec_debug_init, 71
device.c, 72
    ec_device_call_isr, 74
    ec_device_clear, 73
    ec_device_close, 73
    ec_device_init, 72
    ec_device_open, 73
    ec_device_send, 73
    ec_device_tx_data, 73
device.h, 75
    ec_device_call_isr, 76
    ec_device_close, 76
    ec_device_init, 75
    ec_device_open, 76
    ec_device_send, 76
    ec_device_tx_data, 76
DeviceInterface
    ecdev_link_state, 22
    ecdev_receive, 22
    ecdev_register, 23
    ecdev_start, 23
    ecdev_stop, 24
    ecdev_unregister, 23
devices/ Directory Reference, 25
domain.c, 78
    ec_domain_add_datagram, 80
    ec_domain_alloc, 80
    ec_domain_clear, 79
    ec_domain_clear_data_regs, 79
    ec_domain_init, 79
    ec_domain_queue, 81
    ec_domain_reg_pdo_entry, 80
    ec_show_domain_attribute, 79
domain.h, 82
    ec_domain_alloc, 82
    ec_domain_init, 82

EC_ACK
    slave.h, 149
ec_address_t, 29
ec_cleanup_module
    module.c, 139

ec_code_msg_t, 30
ec_data_reg_t, 31
EC_DATAGRAM_APRD
    datagram.h, 65
ec_datagram_aprd
    datagram.c, 61
    datagram.h, 66
EC_DATAGRAM_APWR
    datagram.h, 65
ec_datagram_apwr
    datagram.c, 61
    datagram.h, 67
EC_DATAGRAM_BRD
    datagram.h, 65
ec_datagram_brd
    datagram.c, 62
    datagram.h, 67
EC_DATAGRAM_BWR
    datagram.h, 65
ec_datagram_bwr
    datagram.c, 62
    datagram.h, 67
ec_datagram_clear
    datagram.c, 60
EC_DATAGRAM_ERROR
    datagram.h, 65
EC_DATAGRAM_INIT
    datagram.h, 65
ec_datagram_init
    datagram.c, 60
EC_DATAGRAM_LRW
    datagram.h, 65
ec_datagram_lrw
    datagram.c, 62
    datagram.h, 68
EC_DATAGRAM_NONE
    datagram.h, 65
EC_DATAGRAM_NPRD
    datagram.h, 65
ec_datagram_nprd
    datagram.c, 60
    datagram.h, 66
EC_DATAGRAM_NPWR
    datagram.h, 65
ec_datagram_npwr
    datagram.c, 61
    datagram.h, 66
ec_datagram_prealloc
    datagram.c, 60
    datagram.h, 66
EC_DATAGRAM_QUEUED
    datagram.h, 65
EC_DATAGRAM_RECEIVED
    datagram.h, 65
EC_DATAGRAM_SENT
    datagram.h, 65
ec_datagram_state_t
    datagram.h, 65
ec_datagram_t, 32
EC_DATAGRAM_TIMED_OUT
    datagram.h, 65
ec_datagram_type_t
    datagram.h, 65
EC_DBG
    globals.h, 120
ec_dbgdev_open
    debug.c, 69
ec_dbgdev_stats
    debug.c, 69
ec_dbgdev_stop
    debug.c, 69
ec_debug_clear
    debug.c, 70
    debug.h, 71
ec_debug_init
    debug.c, 70
    debug.h, 71
ec_debug_send
    debug.c, 70
ec_debug_t, 33
ec_device, 34
ec_device_call_isr
    device.c, 74
    device.h, 76
ec_device_clear
    device.c, 73
ec_device_close
    device.c, 73
    device.h, 76
ec_device_init
    device.c, 72
    device.h, 75
ec_device_open
    device.c, 73
    device.h, 76
ec_device_send
    device.c, 73
    device.h, 76
ec_device_t
    ecdev.h, 85
ec_device_tx_data
    device.c, 73
    device.h, 76
ec_domain, 35
ec_domain_add_datagram
    domain.c, 80
ec_domain_alloc
    domain.c, 80

domain.h, 82
ec_domain_clear
domain.c, 79
ec_domain_clear_data_regs
domain.c, 79
ec_domain_init
domain.c, 79
domain.h, 82
ec_domain_queue
domain.c, 81
ec_domain_reg_pdo_entry
domain.c, 80
ec_domain_t
ecrt.h, 92
ec_eoe, 36
ec_eoe_active
ethernet.c, 97
ethernet.h, 99
ec_eoe_clear
ethernet.c, 96
ethernet.h, 99
ec_eoe_flush
ethernet.c, 94
ec_eoe_frame_t, 38
ec_eoe_init
ethernet.c, 96
ethernet.h, 99
ec_eoe_run
ethernet.c, 96
ec_eoe_send
ethernet.c, 96
ec_eoe_state_rx_check
ethernet.c, 94
ec_eoe_state_rx_fetch
ethernet.c, 95
ec_eoe_state_rx_start
ethernet.c, 94
ec_eoe_state_tx_sent
ethernet.c, 95
ec_eoe_state_tx_start
ethernet.c, 95
ec_eoe_t
ethernet.h, 98
ec_eoedev_open
ethernet.c, 95
ec_eoedev_stats
ethernet.c, 96
ec_eoedev_stop
ethernet.c, 95
ec_eoedev_tx
ethernet.c, 95
EC_ERR
globals.h, 120
ec_find_master
module.c, 139
ec_fmmu_config
master.c, 132
master.h, 137
ec_fmmu_t, 39
ec_fsm, 40
ec_fsm_change_ack
fsm.c, 110
ec_fsm_change_check
fsm.c, 110
ec_fsm_change_check_ack
fsm.c, 111
ec_fsm_change_code
fsm.c, 110
ec_fsm_change_start
fsm.c, 110
ec_fsm_change_status
fsm.c, 110
ec_fsm_clear
fsm.c, 112
ec_fsm_coe_down_check
fsm.c, 111
ec_fsm_coe_down_request
fsm.c, 111
ec_fsm_coe_down_response
fsm.c, 111
ec_fsm_coe_down_start
fsm.c, 111
ec_fsm_configuration_conf
fsm.c, 106
ec_fsm_configuration_running
fsm.c, 113
fsm.h, 116
ec_fsm_configuration_start
fsm.c, 106
ec_fsm_configuration_success
fsm.c, 113
fsm.h, 116
ec_fsm_end
fsm.c, 111
ec_fsm_error
fsm.c, 112
ec_fsm_execute
fsm.c, 112
ec_fsm_init
fsm.c, 112
ec_fsm_master_action_addresses
fsm.c, 114
ec_fsm_master_action_next_slave_state
fsm.c, 114
ec_fsm_master_action_process_states
fsm.c, 114
ec_fsm_master_broadcast
fsm.c, 104

ec_fsm_master_configure_slave
fsm.c, 105
ec_fsm_master_read_states
fsm.c, 104
ec_fsm_master_rewrite_addresses
fsm.c, 104
ec_fsm_master_scan_slaves
fsm.c, 105
ec_fsm_master_start
fsm.c, 104
ec_fsm_master_validate_product
fsm.c, 104
ec_fsm_master_validate_vendor
fsm.c, 104
ec_fsm_master_write_eeprom
fsm.c, 105
ec_fsm_reset
fsm.c, 112
ec_fsm_sii_read_check
fsm.c, 109
ec_fsm_sii_read_fetch
fsm.c, 109
ec_fsm_sii_start_reading
fsm.c, 109
ec_fsm_sii_start_writing
fsm.c, 109
ec_fsm_sii_write_check
fsm.c, 109
ec_fsm_sii_write_check2
fsm.c, 110
ec_fsm_slaveconf_fmmu
fsm.c, 108
ec_fsm_slaveconf_init
fsm.c, 107
ec_fsm_slaveconf_op
fsm.c, 108
ec_fsm_slaveconf_preop
fsm.c, 108
ec_fsm_slaveconf_saveop
fsm.c, 108
ec_fsm_slaveconf_sdoconf
fsm.c, 108
ec_fsm_slaveconf_sync
fsm.c, 108
ec_fsm_slavescan_address
fsm.c, 106
ec_fsm_slavescan_base
fsm.c, 107
ec_fsm_slavescan_datalink
fsm.c, 107
ec_fsm_slavescan_eeprom_data
fsm.c, 107
ec_fsm_slavescan_eeprom_size
fsm.c, 107
ec_fsm_slavescan_start
fsm.c, 106
ec_fsm_slavescan_state
fsm.c, 107
ec_fsm_startup_broadcast
fsm.c, 105
ec_fsm_startup_running
fsm.c, 113
fsm.h, 116
ec_fsm_startup_scan
fsm.c, 106
ec_fsm_startup_start
fsm.c, 105
ec_fsm_startup_success
fsm.c, 113
fsm.h, 116
ec_fsm_t
fsm.h, 116
EC_INFO
globals.h, 120
ec_init_module
module.c, 139
EC_LIT
globals.h, 121
ec_master, 42
ec_master_bus_scan
master.c, 130
master.h, 136
ec_master_calc_addressing
master.c, 132
ec_master_clear
master.c, 129
master.h, 135
ec_master_eoe_run
master.c, 128
ec_master_eoe_start
master.c, 132
ec_master_eoe_stop
master.c, 132
ec_master_idle_run
master.c, 128
ec_master_idle_start
master.c, 131
ec_master_idle_stop
master.c, 131
ec_master_info
master.c, 132
ec_master_init
master.c, 129
master.h, 135
ec_master_output_stats
master.c, 131
master.h, 136
ec_master_queue_datagram

master.c, 130
ec_master_receive_datagrams
master.c, 130
master.h, 136
ec_master_reset
master.c, 129
master.h, 135
ec_master_send_datagrams
master.c, 130
ec_master_sync_io
master.c, 128
ec_master_t
ecrt.h, 92
EC_MAX_DATA_SIZE
globals.h, 119
EC_MBOX_AOE
slave.h, 149
EC_MBOX_COE
slave.h, 149
EC_MBOX_EOE
slave.h, 149
EC_MBOX_FOE
slave.h, 149
EC_MBOX_SOE
slave.h, 149
EC_MBOX_VOE
slave.h, 149
ec_pdo_reg_t, 44
ec_print_data
module.c, 140
ec_print_data_diff
module.c, 140
EC_READ_BIT
ecrt.h, 88
EC_READ_S16
ecrt.h, 89
EC_READ_S32
ecrt.h, 90
EC_READ_S8
ecrt.h, 89
EC_READ_U16
ecrt.h, 89
EC_READ_U32
ecrt.h, 89
EC_READ_U8
ecrt.h, 89
EC_RX_PDO
slave.h, 149
ec_sdo_data_t, 45
ec_sdo_entry_t, 46
ec_sdo_t, 47
ec_show_domain_attribute
domain.c, 79
ec_show_master_attribute
master.c, 128
ec_show_slave_attribute
slave.c, 142
ec_sii_pdo_entry_t, 48
ec_sii_pdo_t, 49
ec_sii_pdo_type_t
slave.h, 149
ec_sii_string_t, 50
ec_sii_sync_t, 51
ec_slave, 52
sii_group, 55
sii_image, 55
sii_name, 55
sii_order, 55
ec_slave_calc_sync_size
slave.c, 145
slave.h, 151
ec_slave_clear
slave.c, 143
ec_slave_conf_sdo
slave.c, 146
ec_slave_fetch_general
slave.c, 143
slave.h, 150
ec_slave_fetch_pdo
slave.c, 144
slave.h, 151
ec_slave_fetch_strings
slave.c, 143
slave.h, 150
ec_slave_fetch_sync
slave.c, 143
slave.h, 150
ec_slave_info
slave.c, 145
ec_slave_init
slave.c, 142
slave.h, 149
ec_slave_is_coupler
slave.c, 146
slave.h, 152
ec_slave_locate_string
slave.c, 144
slave.h, 151
ec_slave_mbox_check
mailbox.c, 123
mailbox.h, 125
ec_slave_mbox_fetch
mailbox.c, 123
mailbox.h, 125
ec_slave_mbox_prepare_check
mailbox.c, 122
mailbox.h, 124
ec_slave_mbox_prepare_fetch

mailbox.c, 123
mailbox.h, 125
ec_slave_mbox_prepare_send
mailbox.c, 122
mailbox.h, 124
ec_slave_prepare_fmmu
slave.c, 144
slave.h, 149
EC_SLAVE_STATE_INIT
slave.h, 148
EC_SLAVE_STATE_OP
slave.h, 149
EC_SLAVE_STATE_PREOP
slave.h, 149
EC_SLAVE_STATE_SAVEOP
slave.h, 149
ec_slave_state_t
slave.h, 148
EC_SLAVE_STATE_UNKNOWN
slave.h, 148
ec_slave_t
ecrt.h, 92
ec_slave_write_eeprom
slave.c, 145
ec_state_string
module.c, 140
ec_stats_t, 56
ec_store_master_attribute
master.c, 129
ec_store_slave_attribute
slave.c, 142
EC_STR
globals.h, 121
ec_sync_config
master.c, 131
master.h, 136
EC_SYSFS_READ_ATTR
globals.h, 121
EC_SYSFS_READ_WRITE_ATTR
globals.h, 121
EC_TX_PDO
slave.h, 149
ec_varsize_t, 57
EC_WARN
globals.h, 120
EC_WRITE_BIT
ecrt.h, 88
EC_WRITE_S16
ecrt.h, 91
EC_WRITE_S32
ecrt.h, 91
EC_WRITE_S8
ecrt.h, 90
EC_WRITE_U16
ecrt.h, 90
EC_WRITE_U32
ecrt.h, 91
EC_WRITE_U8
ecrt.h, 90
ecdb.h, 84
ecdev.h, 85
ec_device_t, 85
ecdev_link_state
DeviceInterface, 22
ecdev_receive
DeviceInterface, 22
ecdev_register
DeviceInterface, 23
ecdev_start
DeviceInterface, 23
ecdev_stop
DeviceInterface, 24
ecdev_unregister
DeviceInterface, 23
ecrt.h, 86
ec_domain_t, 92
ec_master_t, 92
EC_READ_BIT, 88
EC_READ_S16, 89
EC_READ_S32, 90
EC_READ_S8, 89
EC_READ_U16, 89
EC_READ_U32, 89
EC_READ_U8, 89
ec_slave_t, 92
EC_WRITE_BIT, 88
EC_WRITE_S16, 91
EC_WRITE_S32, 91
EC_WRITE_S8, 90
EC_WRITE_U16, 90
EC_WRITE_U32, 91
EC_WRITE_U8, 90
ecrt_domain_process
RealtimeInterface, 17
ecrt_domain_register_pdo
RealtimeInterface, 16
ecrt_domain_register_pdo_list
RealtimeInterface, 17
ecrt_domain_state
RealtimeInterface, 17
ecrt_master_activate
RealtimeInterface, 18
ecrt_master_callbacks
RealtimeInterface, 19
ecrt_master_create_domain
RealtimeInterface, 17
ecrt_master_deactivate
RealtimeInterface, 18

ecrt_master_get_slave
RealtimeInterface, 19
ecrt_master_prepare
RealtimeInterface, 18
ecrt_master_receive
RealtimeInterface, 18
ecrt_master_run
RealtimeInterface, 19
ecrt_master_send
RealtimeInterface, 18
ecrt_release_master
RealtimeInterface, 20
ecrt_request_master
RealtimeInterface, 20
ecrt_slave_conf_sdo16
RealtimeInterface, 20
ecrt_slave_conf_sdo32
RealtimeInterface, 21
ecrt_slave_conf_sdo8
RealtimeInterface, 20
ecrt_slave_pdo_size
RealtimeInterface, 21
EOE_DEBUG_LEVEL
ethernet.c, 94
EtherCAT device interface, 22
EtherCAT realtime interface, 15
ethernet.c, 93
ec_eoe_active, 97
ec_eoe_clear, 96
ec_eoe_flush, 94
ec_eoe_init, 96
ec_eoe_run, 96
ec_eoe_send, 96
ec_eoe_state_rx_check, 94
ec_eoe_state_rx_fetch, 95
ec_eoe_state_rx_start, 94
ec_eoe_state_tx_sent, 95
ec_eoe_state_tx_start, 95
ec_eoedev_open, 95
ec_eoedev_stats, 96
ec_eoedev_stop, 95
ec_eoedev_tx, 95
EOE_DEBUG_LEVEL, 94
ethernet.h, 98
ec_eoe_active, 99
ec_eoe_clear, 99
ec_eoe_init, 99
ec_eoe_t, 98

fsm.c, 100
ec_fsm_change_ack, 110
ec_fsm_change_check, 110
ec_fsm_change_check_ack, 111
ec_fsm_change_code, 110
ec_fsm_change_start, 110
ec_fsm_change_status, 110
ec_fsm_clear, 112
ec_fsm_coe_down_check, 111
ec_fsm_coe_down_request, 111
ec_fsm_coe_down_response, 111
ec_fsm_coe_down_start, 111
ec_fsm_configuration_conf, 106
ec_fsm_configuration_running, 113
ec_fsm_configuration_start, 106
ec_fsm_configuration_success, 113
ec_fsm_end, 111
ec_fsm_error, 112
ec_fsm_execute, 112
ec_fsm_init, 112
ec_fsm_master_action_addresses, 114
ec_fsm_master_action_next_slave_state, 114
ec_fsm_master_action_process_states, 114
ec_fsm_master_broadcast, 104
ec_fsm_master_configure_slave, 105
ec_fsm_master_read_states, 104
ec_fsm_master_rewrite_addresses, 104
ec_fsm_master_scan_slaves, 105
ec_fsm_master_start, 104
ec_fsm_master_validate_product, 104
ec_fsm_master_validate_vendor, 104
ec_fsm_master_write_eeprom, 105
ec_fsm_reset, 112
ec_fsm_sii_read_check, 109
ec_fsm_sii_read_fetch, 109
ec_fsm_sii_start_reading, 109
ec_fsm_sii_start_writing, 109
ec_fsm_sii_write_check, 109
ec_fsm_sii_write_check2, 110
ec_fsm_slaveconf_fmmu, 108
ec_fsm_slaveconf_init, 107
ec_fsm_slaveconf_op, 108
ec_fsm_slaveconf_preop, 108
ec_fsm_slaveconf_saveop, 108
ec_fsm_slaveconf_sdoconf, 108
ec_fsm_slaveconf_sync, 108
ec_fsm_slavescan_address, 106
ec_fsm_slavescan_base, 107
ec_fsm_slavescan_datalink, 107
ec_fsm_slavescan_eeprom_data, 107
ec_fsm_slavescan_eeprom_size, 107
ec_fsm_slavescan_start, 106
ec_fsm_slavescan_state, 107
ec_fsm_startup_broadcast, 105
ec_fsm_startup_running, 113
ec_fsm_startup_scan, 106
ec_fsm_startup_start, 105
ec_fsm_startup_success, 113
sdo_abort_messages, 114

fsm.h, 115
ec_fsm_configuration_running, 116
ec_fsm_configuration_success, 116
ec_fsm_startup_running, 116
ec_fsm_startup_success, 116
ec_fsm_t, 116

globals.h, 118
EC_DBG, 120
EC_ERR, 120
EC_INFO, 120
EC_LIT, 121
EC_MAX_DATA_SIZE, 119
EC_STR, 121
EC_SYSFS_READ_ATTR, 121
EC_SYSFS_READ_WRITE_ATTR, 121
EC_WARN, 120

include/ Directory Reference, 26

mailbox.c, 122
ec_slave_mbox_check, 123
ec_slave_mbox_fetch, 123
ec_slave_mbox_prepare_check, 122
ec_slave_mbox_prepare_fetch, 123
ec_slave_mbox_prepare_send, 122
mailbox.h, 124
ec_slave_mbox_check, 125
ec_slave_mbox_fetch, 125
ec_slave_mbox_prepare_check, 124
ec_slave_mbox_prepare_fetch, 125
ec_slave_mbox_prepare_send, 124
master.c, 126
ec_fmmu_config, 132
ec_master_bus_scan, 130
ec_master_calc_addressing, 132
ec_master_clear, 129
ec_master_eoe_run, 128
ec_master_eoe_start, 132
ec_master_eoe_stop, 132
ec_master_idle_run, 128
ec_master_idle_start, 131
ec_master_idle_stop, 131
ec_master_info, 132
ec_master_init, 129
ec_master_output_stats, 131
ec_master_queue_datagram, 130
ec_master_receive_datagrams, 130
ec_master_reset, 129
ec_master_send_datagrams, 130
ec_master_sync_io, 128
ec_show_master_attribute, 128
ec_store_master_attribute, 129
ec_sync_config, 131

master.h, 134
ec_fmmu_config, 137
ec_master_bus_scan, 136
ec_master_clear, 135
ec_master_init, 135
ec_master_output_stats, 136
ec_master_receive_datagrams, 136
ec_master_reset, 135
ec_sync_config, 136
master/ Directory Reference, 27
module.c, 138
COMPILE_INFO, 139
ec_cleanup_module, 139
ec_find_master, 139
ec_init_module, 139
ec_print_data, 140
ec_print_data_diff, 140
ec_state_string, 140

RealtimeInterface
ecrt_domain_process, 17
ecrt_domain_register_pdo, 16
ecrt_domain_register_pdo_list, 17
ecrt_domain_state, 17
ecrt_master_activate, 18
ecrt_master_callbacks, 19
ecrt_master_create_domain, 17
ecrt_master_deactivate, 18
ecrt_master_get_slave, 19
ecrt_master_prepare, 18
ecrt_master_receive, 18
ecrt_master_run, 19
ecrt_master_send, 18
ecrt_release_master, 20
ecrt_request_master, 20
ecrt_slave_conf_sdo16, 20
ecrt_slave_conf_sdo32, 21
ecrt_slave_conf_sdo8, 20
ecrt_slave_pdo_size, 21

sdo_abort_messages
fsm.c, 114
sii_group
ec_slave, 55
sii_image
ec_slave, 55
sii_name
ec_slave, 55
sii_order
ec_slave, 55
slave.c, 141
ec_show_slave_attribute, 142
ec_slave_calc_sync_size, 145
ec_slave_clear, 143

ec_slave_conf_sdo, 146
ec_slave_fetch_general, 143
ec_slave_fetch_pdo, 144
ec_slave_fetch_strings, 143
ec_slave_fetch_sync, 143
ec_slave_info, 145
ec_slave_init, 142
ec_slave_is_coupler, 146
ec_slave_locate_string, 144
ec_slave_prepare_fmmu, 144
ec_slave_write_eeprom, 145
ec_store_slave_attribute, 142
slave.h, 147
EC_ACK, 149
EC_MBOX_AOE, 149
EC_MBOX_COE, 149
EC_MBOX_EOE, 149
EC_MBOX_FOE, 149
EC_MBOX_SOE, 149
EC_MBOX_VOE, 149
EC_RX_PDO, 149
ec_sii_pdo_type_t, 149
ec_slave_calc_sync_size, 151
ec_slave_fetch_general, 150
ec_slave_fetch_pdo, 151
ec_slave_fetch_strings, 150
ec_slave_fetch_sync, 150
ec_slave_init, 149
ec_slave_is_coupler, 152
ec_slave_locate_string, 151
ec_slave_prepare_fmmu, 149
EC_SLAVE_STATE_INIT, 148
EC_SLAVE_STATE_OP, 149
EC_SLAVE_STATE_PREOP, 149
EC_SLAVE_STATE_SAVEOP, 149
ec_slave_state_t, 148
EC_SLAVE_STATE_UNKNOWN, 148
EC_TX_PDO, 149