eclib manual

version 0.1

M.J.C. Ronde
m.j.c.ronde@tue.nl

June 6, 2011

Prerequisites

To use this library you need SuperUser rights on your system and have SOEM preinstalled.

Glossary:

link_id:	Counter for a specific module in the EtherCAT stack connected. Starts at 0.
chan_id:	Counter for channels on a specific module.
port_id:	Logical counter for a specific port type.

Return codes:

```
#define EC_SUCCESS                  0
#define EC_ERR_SOCKET              -1
#define EC_ERR_OPSTATE             -2
#define EC_ERR_NOSLAVES            -3
#define EC_ERR_INVALID_LINK_ID     -4
#define EC_ERR_INVALID_CHAN_ID     -5
#define EC_ERR_IO                  -6
#define EC_ERR_INVALID_PORT_ID     -7
```

Function for starting/stopping/io

```
int ec_start(char* netif);
```

Start the software with EtherCAT stack connected to netif, e.g. ec_start (”eth0”).

```
int ec_io(void);
```

Refreshes in- and output values of the EtherCAT stack, i.e. the input values are refreshed and output values written to the mapping are made effective at the output. Must be done at a rate of 10 Hz or higher, otherwise the EtherCAT stack will go to SAFE_OP.

```
int ec_stop(void);
```

Stop the software.

```
void ec_print_error_msg(int retval);
```

Print error message related to the returnvalue retval of another library function

Module specific functions:

Read functions

```
int ec_EL1008_di_read_chan(double *pvalue, int chan_id, int link_id);
int ec_EL1014_di_read_chan(double *pvalue, int chan_id, int link_id);
int ec_EL1018_di_read_chan(double *pvalue, int chan_id, int link_id);

int ec_EL3102_adc_read_chan(double *pvalue, int chan_id, int link_id);

int ec_EL5101_enc_read_chan(double *pvalue, int link_id);
```

All read functions **int** ec_X_Y_read_chan(**double** *pvalue, **int** chan_id, **int** link_id);, where *X* is the module name and *Y* is the type of channel which is read (adc, di, enc). The arguments of these functions are a pointer to the address to store the value, chan_id (range depends on the module) and link_id.

```
int ec_EL2004_do_write_chan(double value, int chan_id, int link_id);
int ec_EL2008_do_write_chan(double value, int chan_id, int link_id);

int ec_EL4038_dac_write_chan(double value, int chan_id, int link_id);
int ec_EL4132_dac_write_chan(double value, int chan_id, int link_id);
```

All write functions **int** ec_X_Y_read_chan(**double** value, **int** chan_id, **int** link_id);, where *X* is the module name and *Y* is the type of channel which is writen (do, dac). The arguments of these functions are the value to write (allowable range depends on module), chan_id (range depends on the module) and link_id.

E-box read

```
int ec_Ebox_adc_read_chan(double *pvalue, int chan_id, int link_id);
int ec_Ebox_enc_read_chan(double *pvalue, int chan_id, int link_id);
int ec_Ebox_di_read_chan(double *pvalue, int chan_id, int link_id);
```

E-box write

```
int ec_Ebox_dac_write_chan(double pvalue, int chan_id, int link_id);
int ec_Ebox_do_write_chan(double pvalue, int chan_id, int link_id);
int ec_Ebox_pwm_write_chan(double pvalue, int chan_id, int link_id);
```

Logical port functions:

```
int ec_adc_read_chan(double *pvalue, int port_id);
int ec_adc_get_clipped_port(int *pvalue, int port_id);
int ec_enc_read_chan(double *pvalue, int port_id);
int ec_din_read_chan(double *pvalue, int port_id);

int ec_dac_write_chan(double value, int port_id);
int ec_dout_write_chan(double value, int port_id);
int ec_pwm_write_chan(double value, int port_id);
```

Functions to retrieve the number of

```
int ec_Ebox_get_ndevs(void);
```

Returns the number of E-box connected.
The following functions return the number of ports of a specific type of the connected EtherCAT stack.

```c
int ec_get_nadc(void);
int ec_get_ndac(void);
int ec_get_nenc(void);
int ec_get_ndin(void);
int ec_get_ndout(void);
int ec_get_npwm(void);
```