EtherCat

- Faulhaber benutzt CanOpen over EtherCAT
    - EtherCAT ist nur delivery mechanism
    - Versendete Daten sind CanOopen kompatibel
- Es wird CAN in Automation (CiA) und CiA 302 eingesetzt

Handbuch	Beschreibung
CiA 301	CANopen application layer and communication profile
CiA 402	CANopen device profile for drives and motion control

CiA 402 Statemachine

```
Error parsing Mermaid diagram!

Cannot read properties of null (reading 'getBBox')
```

Standard vorgehen Motion Controller

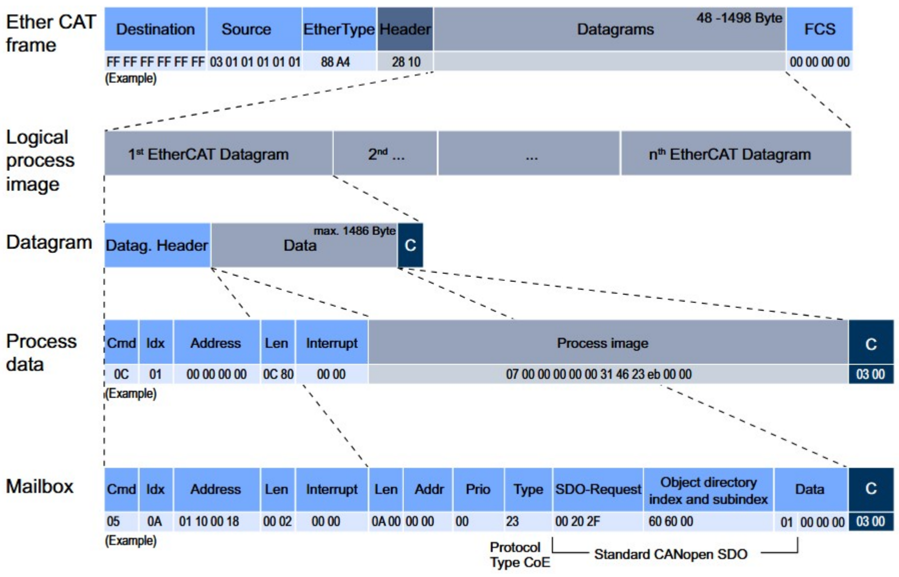
```
Error parsing Mermaid diagram!

Cannot read properties of null (reading 'getBBox')
```

Objektverzeichnis

Das Objektverzeichnis verwaltet die Konfigurationsparameter. Das Objektverzeichnis ist in drei Bereiche unterteilt. Jedes Objekt kann über seinen Index und Subindex referenziert werden (SDO-Protokoll).

- Kommunikationsparameter (Index 0x1000 bis 0x1FFF, enthält Kommunikationsobjekte nach CiA 301, siehe Kap. 4.1, S. 36)
- Herstellerspezifischer Bereich (Index 0x2000 bis 0x5FFF, enthält herstellerspezifische Objekte, siehe Kap. 4.2, S. 44)
- Standardisierte Geräteprofile (0x6000 bis 0x9FFF, enthält die vom Motion Controller unterstützten Objekte, siehe Dokumentation der Antriebsfunktionen)

Index	Zuordnung der Objekte
x1000 - 0x1FFF	Kommunikationsobjekte
0x2000 - 0x5FFF	Herstellerspezifische Objekte
0x6000 - 0x6FFF	Objekte des Antriebsprofils nach CiA 402



Control Word

The control object 0x6040 is used to request state changes on the CiA402 state machine. The control commands are coded by bits.

Bit	Description
0	Switch on
1	Enable voltage
2	Quick Stop
3	Enable Operation
4-6	Operation mode specific
7	Fault reset

These defines are available to transition between different states.

Define	Value	Description
CONTROLWORD_COMMAND_SHUTDOWN	0x0006	Shutdown command
CONTROLWORD_COMMAND_SWITCHON	0x0007	Switch on command
CONTROLWORD_COMMAND_SWITCHON_ENABLEOPERATION	0x000F	Switch on and enable command
CONTROLWORD_COMMAND_DISABLEVOLTAGE	0x0000	Disable voltage command
CONTROLWORD_COMMAND_QUICKSTOP	0x0002	Quickstop command
CONTROLWORD_COMMAND_DISABLEOPERATION	0x0007	Disable operation command
CONTROLWORD_COMMAND_ENABLEOPERATION	0x000F	Enable operation command
CONTROLWORD_COMMAND_FAULTRESET	0x0080	Fault reset command

Status Word

The status object 0x6041 is used to get the state of the CiA402 state machine.

Bit	Description
0	Ready to switch on
1	Switched on
2	Operation enabled
3	Fault
4	Voltage enabled
5	Quick Stop
6	Switch on disabled

These defines are available to interpret the FSA states.

Define	Value	Description
STATUSWORD_STATE_NOTREADYTOSWITCHON	0x0000	Not ready to switch on
STATUSWORD_STATE_SWITCHEDONDISABLED	0x0040	Switched on but disabled
STATUSWORD_STATE_READYTOSWITCHON	0x0021	Ready to switch on
STATUSWORD_STATE_SWITCHEDON	0x0023	Switched on
STATUSWORD_STATE_OPERATIONENABLED	0x0027	Operation enabled
STATUSWORD_STATE_QUICKSTOPACTIVE	0x0007	Quickstop active
STATUSWORD_STATE_FAULTREACTIONACTIVE	0x000F	Fault reaction active
STATUSWORD_STATE_FAULT	0x0008	Fault state

Basic Setup SOEM

1. Initialize SOEM with Context

```
ecx_contextt ecx_context; // your EtherCAT context

if (ecx_init(&ecx_context, "eth0")) {

    ecx_config_init(&ecx_context, FALSE);

    ecx_config_map(&ecx_context, &IOmap);

    ecx_configdc(&ecx_context);

}
```

2. Set Operation Mode (e.g., Profile Position Mode)

```
uint8 mode = 1; // Profile Position Mode

ecx_SDOwrite(&ecx_context, slave, 0x6060, 0x00, FALSE, sizeof(mode), &mode, EC_TIMEOUTRXM);
```

3. **Transition Through CiA 402 States Using Control Word (0x6040)**

a. Enable Voltage (`0x06`)

```
uint16 control_word = 0x0006;

ecx_SDOwrite(&ecx_context, slave, 0x6040, 0x00, FALSE, sizeof(control_word), &control_word, EC_TIMEOUTRXM);
```

b. Switch On (`0x07`)

```
control_word = 0x0007;

ecx_SDOwrite(&ecx_context, slave, 0x6040, 0x00, FALSE, sizeof(control_word), &control_word, EC_TIMEOUTRXM);
```

c. Enable Operation (`0x0F`)

```
control_word = 0x000F;

ecx_SDOwrite(&ecx_context, slave, 0x6040, 0x00, FALSE, sizeof(control_word), &control_word, EC_TIMEOUTRXM);
```

4. Write Target Position (0x607A)

```
int32 target_position = 10000; // example position

ecx_SDOwrite(&ecx_context, slave, 0x607A, 0x00, FALSE, sizeof(target_position), &target_position, EC_TIMEOUTRXM);
```

5. Start Motion (e.g., set bit 4 in Control Word)

```
control_word = 0x003F; // Enable operation + new set-point

ecx_SDOwrite(&ecx_context, slave, 0x6040, 0x00, FALSE, sizeof(control_word), &control_word, EC_TIMEOUTRXM);
```

6. Read Status Word (0x6041)

```
uint16 status_word;

int size = sizeof(status_word);

ecx_SDOread(&ecx_context, slave, 0x6041, 0x00, FALSE, &size, &status_word, EC_TIMEOUTRXM);
```

SOEM CoE

Read

```
int ecx_SDOread(ecx_contextt *context, uint16 slave, uint16 index, uint8 subindex, boolean CA, int *psize, void *p,
int timeout)
```

```
CoE SDO read, blocking. Single subindex or Complete Access.
```

Only a “normal” upload request is issued. If the requested parameter is <= 4bytes then a “expedited” response is returned, otherwise a “normal” response. If a “normal” response is larger than the mailbox size then the response is segmented. The function will combine all segments and copy them to the parameter buffer.

Parameters:

- **context** – **[in]** context struct
- **slave** – **[in]** Slave number
- **index** – **[in]** Index to read
- **subindex** – **[in]** Subindex to read, must be 0 or 1 if CA is used.
- **CA** – **[in]** FALSE = single subindex. TRUE = Complete Access, all subindexes read.
- **psize** – **[inout]** Size in bytes of parameter buffer, returns bytes read from SDO.
- **p** – **[out]** Pointer to parameter buffer
- **timeout** – **[in]** Timeout in us, standard is EC_TIMEOUTRXM
  Returns:
- Workcounter from last slave response

Write

```
int ecx_SDOwrite(ecx_contextt *context, uint16 Slave, uint16 Index, uint8 SubIndex, boolean CA, int psize, const
void *p, int Timeout)
```

```
CoE SDO write, blocking. Single subindex or Complete Access.
```

A “normal” download request is issued, unless we have small data, then a “expedited” transfer is used. If the parameter is larger than the mailbox size then the download is segmented. The function will split the parameter data in segments and send them to the slave one by one.

Parameters:

- **context** – **[in]** context struct

- **Slave** – **[in]** Slave number
- **Index** – **[in]** Index to write
- **SubIndex** – **[in]** Subindex to write, must be 0 or 1 if CA is used.
- **CA** – **[in]** FALSE = single subindex. TRUE = Complete Access, all subindexes written.
- **psize** – **[in]** Size in bytes of parameter buffer.
- **p** – **[out]** Pointer to parameter buffer
- **Timeout** – **[in]** Timeout in us, standard is EC_TIMEOUTRXM

  Returns:
- Workcounter from last slave response