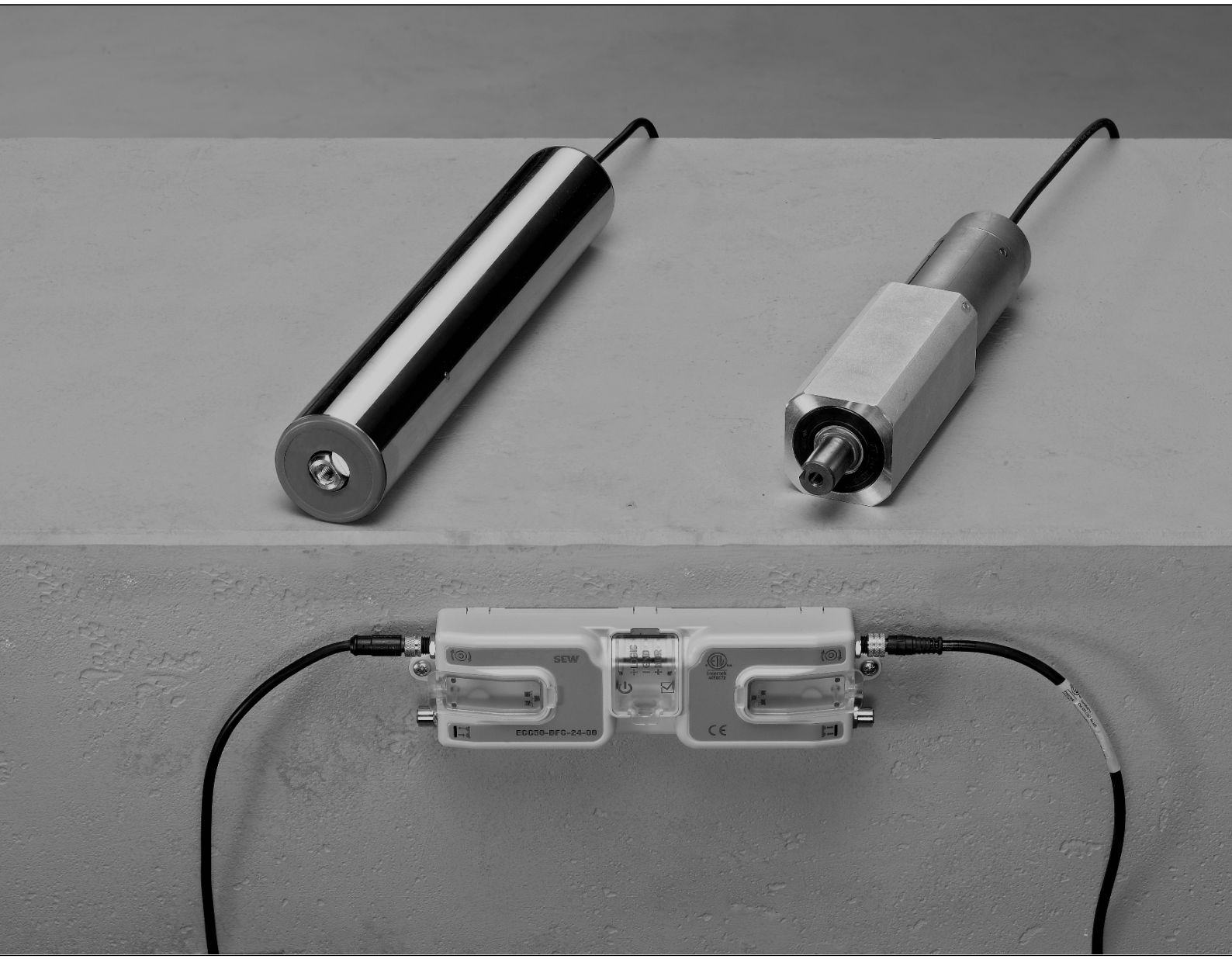
# Addendum to the Operating Instructions




ECDriveS®

Fieldbus Controller ECC-DFC-24-10

Date 10/2020

26856239/EN

**Table of contents**

# 1. General information

INFORMATION

This addendum to the operating instructions contains additional information for the version "Fieldbus controller ECC-DFC-24-10".

Use the data specified in this addendum. This document does not replace the detailed operating instructions "ECDriveS® ECC-DFC fieldbus controller, ECR roller drives, ECG gearmotor".

# 2. Description of the fieldbus controller

The fieldbus controller ECC-DFC-24-10 is a variant of the ECC-DFC-24-00 fieldbus controller. It differs from the standard version by the following functions.

| Function | ECC-DFC-24-00 | ECC-DFC-24-10 |
|---|---|---|
| Fieldbus | Ethernet/IP, PROFINET IO®, Modbus TCP | EtherCAT® |
| Integrated application module | Zero Pressure Accumulation (ZPA) | Not available |
| Engineering Software | ECShell | EtherCAT® Shell |
| Look Ahead Function | Implemented | Not implemented |
| Fire mode | Implemented | Not implemented |
| Binary mode | Implemented | Not implemented |
| PLC functionality | Freely programmable via ECLogix | Not programmable |

## 2.1. Type code

ECC-DFC-24-10: Fieldbus module, EtherCAT®

# 3. Startup

## 3.1. Startup procedure

- Startup takes place exclusively in the EtherCAT® Master Integration of the device description file (ESI file)
- Parameterization
- Configuration of further process data objects (optional)

## 3.2. Supported services

### 3.2.1. File Over EtherCat® (FOE)

With the FOE service, it is possible to perform a firmware update or an update of the $E^2$PROM on a slave device via the bus master.

### 3.2.2. Ethernet Over EtherCAT® (EOE)

Other Ethernet-based protocols and services can also be used with EtherCAT® technology.

Ethernet frames are tunneled via the EtherCAT® protocol. Doing so does not affect the real-time function.

### 3.2.3. Can Over EtherCAT® (CoE)

EtherCAT® provides the communication mechanisms known from CANopen:

- Object list
- Process data objects (PDO)
- Service data objects (SDO)

CoE is the most widespread acyclic EtherCAT® communication. The object directory of a slave is accessed via SDOs. The master can access the parameters both at startup and during operation. CoE offers configuration mechanisms of the PDOs (process data objects) for cyclic communication, such as process data length and process data content. This is useful when configuring slaves with variable PDOs, such as gateways and bus couplers.

## 3.3. Device description files

### 3.3.1. EtherCAT® Slave Information File (ESI file)

The ESI file is a device description in XML format. The user creates this file in the configuration tool separately for each EtherCAT® slave. The ESI file contains information about the objects and object properties of the slaves. The configuration tool accesses the ESI files to create the ENI file.

### 3.3.2. EtherCAT® Network Information File (ENI file)

The ENI file is created by the configuration tool. The ENI file provides the EtherCAT® master with the following information:

- Information on the network topology
- Information on the initialization commands for all network stations
- Commands that the master sends cyclically to the slaves

### 3.3.3. Slave Information Interface (SII)

SII is an $E^2$PROM that contains information about the hardware configuration of the EtherCAT® slave controller (ESC). When booting, the slave applies the SII file to the register of the ESC.

## 3.4. Integration of ECC-DFC-24-10 under an EtherCat® master

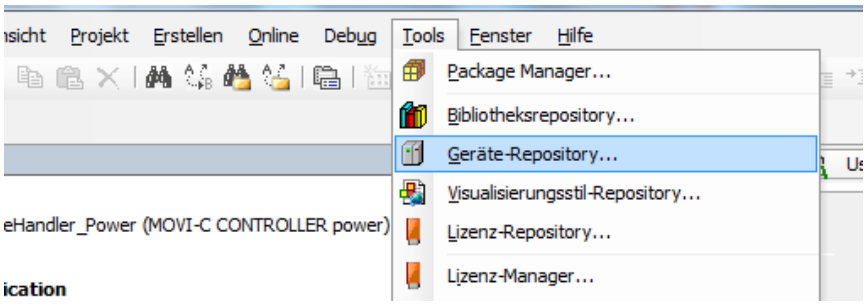
### 3.4.1. Integration of ECC-DFC-24-10 in CODESYS

The procedure described below is an example and refers to version v3.2, SP14, Patch3

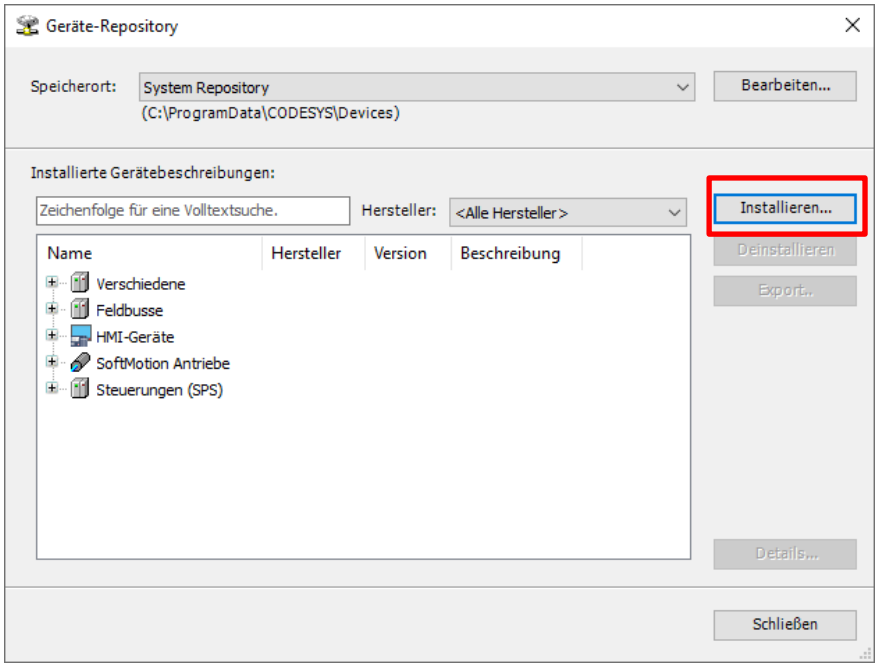
#### Installation of the ESI file

In the IEC Editor, the ESI file can be imported via the Device Repository:

1. Open the "Device Repository" tool.



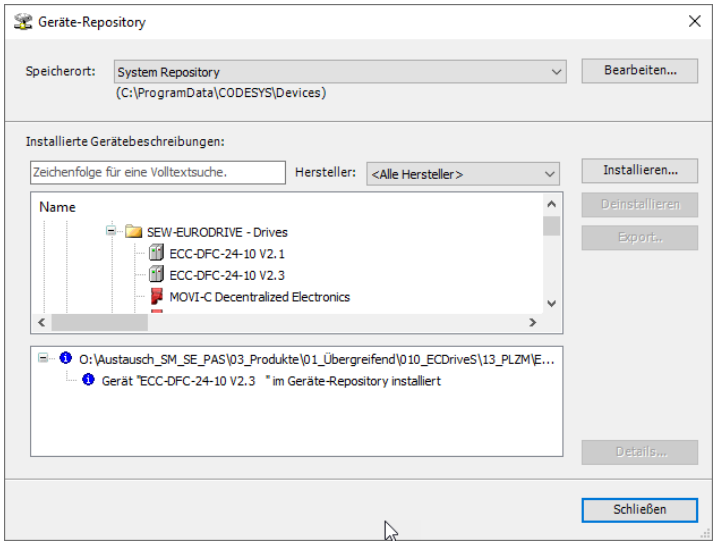
2. In the Device Repository, click the "Install" button.




3. Select the file you saved locally in Explorer and then click "Open".

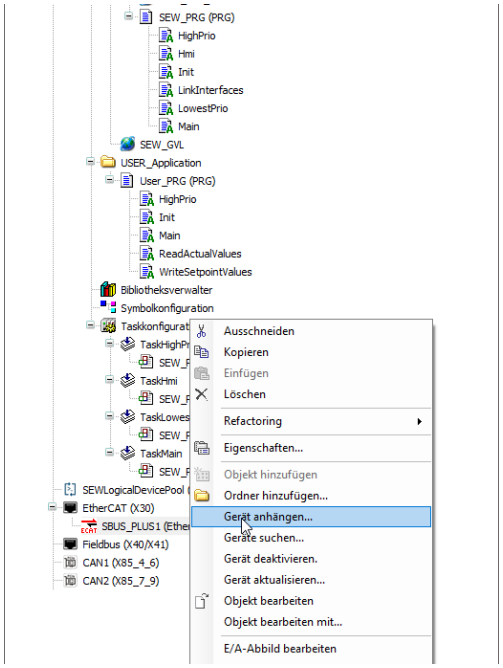
## INFORMATION

After the ESI file has been successfully installed, the ECC-DFC-24-10 module appears in the Device Repository under Feldbusse/EtherCAT/Slaves/ SEW-EURODRIVE - Drives.

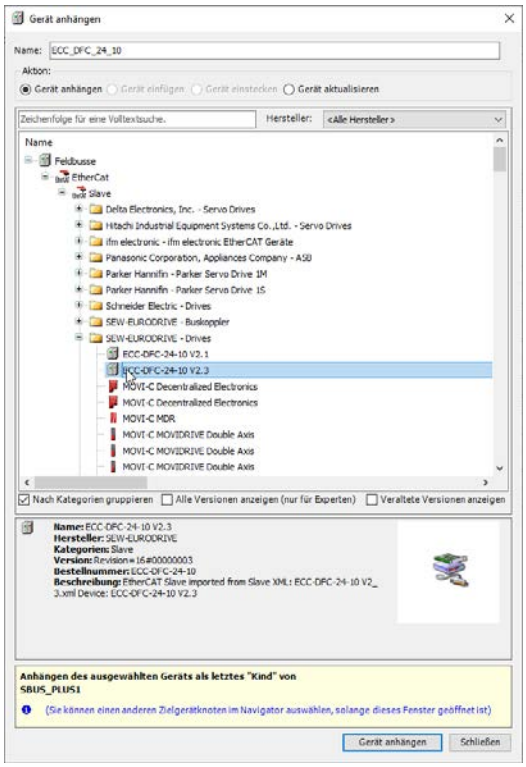



## Manual integration into the project

1. Right-click the EtherCAT® Master in the device view and select the "Attach Device" function.



2. Select the ECDriveS® fieldbus module from the catalog and confirm your selection by pressing the "Attach Device" button

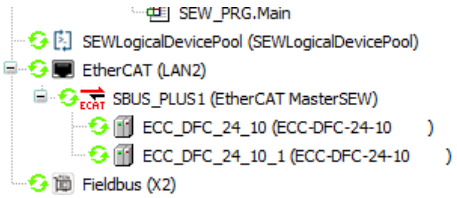



### Automatic integration into the project

Right-click the EtherCAT® Master in the device view and select the "Search for Device" function.

INFORMATION

If the integration into the project was successful, two green rotating arrows will appear next to the device after switching to online mode



### 3.4.2. Integration of ECC-DFC-24-00 in a Beckhoff TwinCAT® project

The procedure described below is an example and refers to version TwinCAT® 3 v2013

#### Installation of ESI files

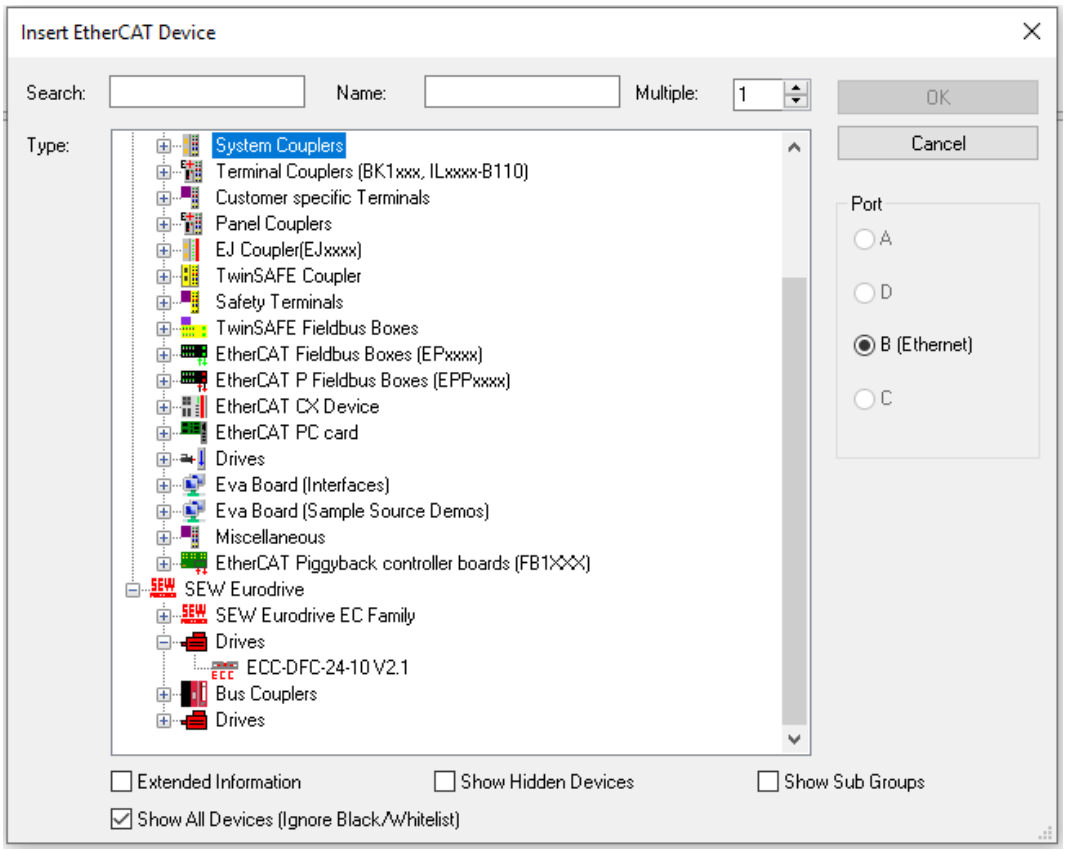
Copy the ESI file to the folder provided for this purpose. This is located in the TwinCAT® folder structure. Usually, the file path to the matching folder looks like this:

C:\TwinCAT\3.1\Config\IO\EtherCAT

#### Manual integration into the project

It is possible to integrate a device into the project offline.

1. To do this, right-click "Device" in the IO configuration. In the shortcut menu, select the "Add New Item" function.
2. Select the ECC-DFC-24-10 module from the catalog and click "OK".



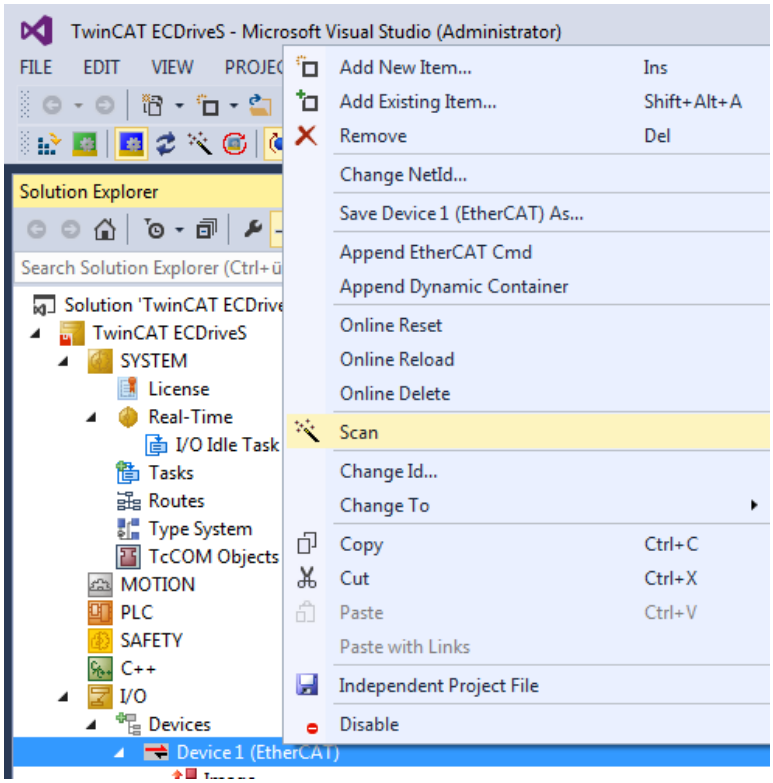

#### Automatic integration into the project

TwinCAT® also offers the possibility of scanning the physical periphery of the EtherCAT® phase and automatically integrating a device into the project.

1. Right-click "Device" in the IO configuration. In the shortcut menu, select the "Scan" function.

INFORMATION

This method is only available if the device has already been correctly connected and is ready for operation. The scan function is only available in "Config Mode"




### 3.4.3. Standard process data objects

After the fieldbus modules ECC-DFC-24-10 have been successfully integrated into the project, the devices appear in the IO configuration. The following process data objects are available by default

Overview of process data objects

| Object type | Designation | Description |
|---|---|---|
| Input data | Inputs 0 | Sensor status, status values of the positioning function |

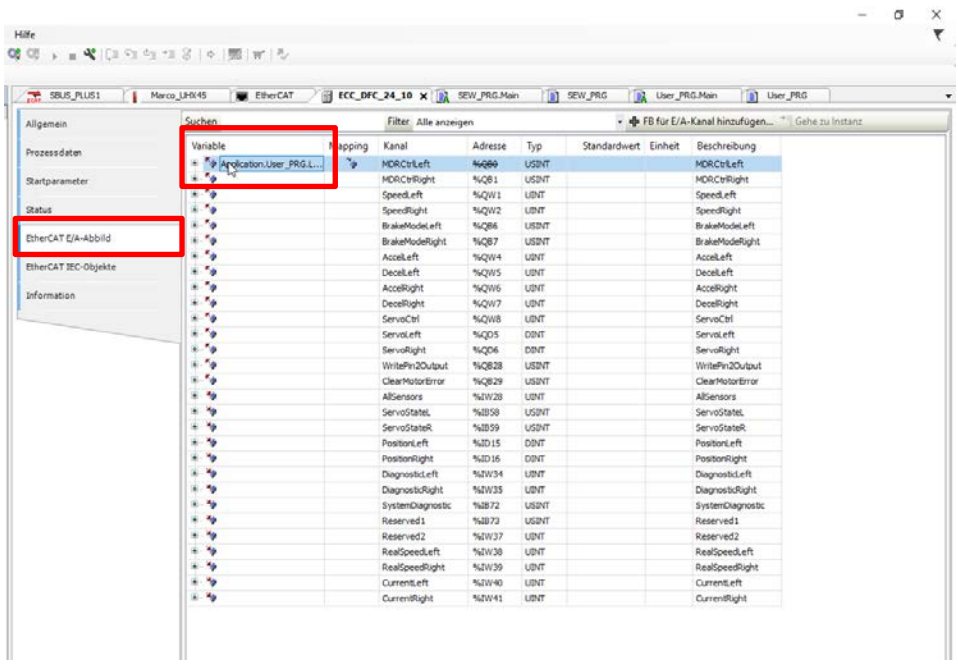
| Object type | Designation | Description |
|---|---|---|
| | Inputs 1 | Diagnostics |
| | Inputs 2 | Actual speed, actual current |
| Output data | Outputs 1 | Motor control words, setpoint speed, braking mode |
| | Outputs 2 | Acceleration / deceleration ramps |
| | Outputs 3 | Positioning function |

### 3.4.4. Assignment of process data

The contents of the process data objects (process data) can be linked with variables of the user program.

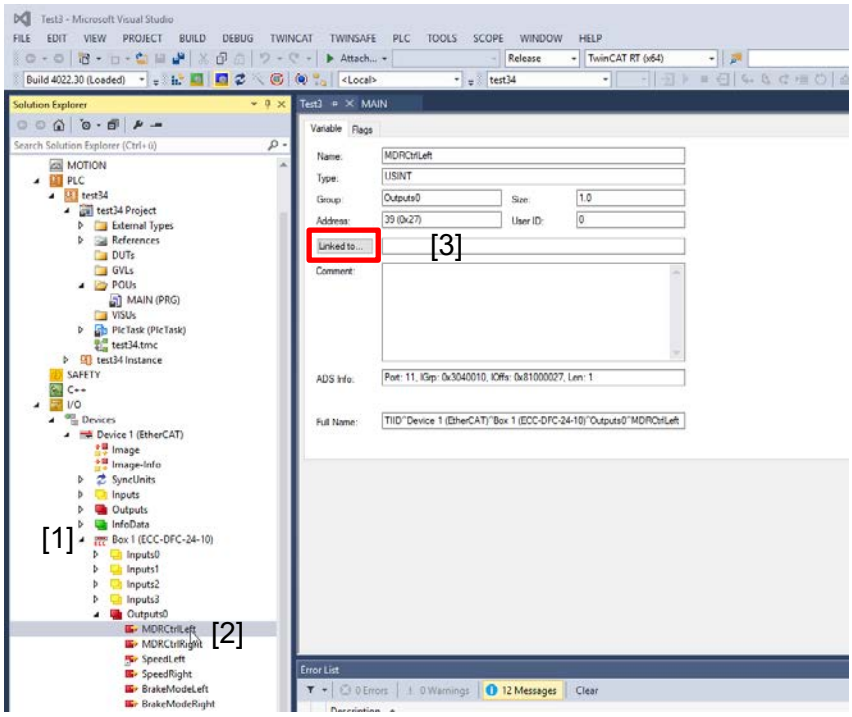
#### Assignment of process data in CODESYS

1. Open the main menu of the corresponding module by double-clicking it.
2. Select the "EtherCAT® I/O Image" submenu. The process data structure is displayed.
3. In the "Variable" field, you can now assign an already defined variable. This variable must correspond to the type of the object.

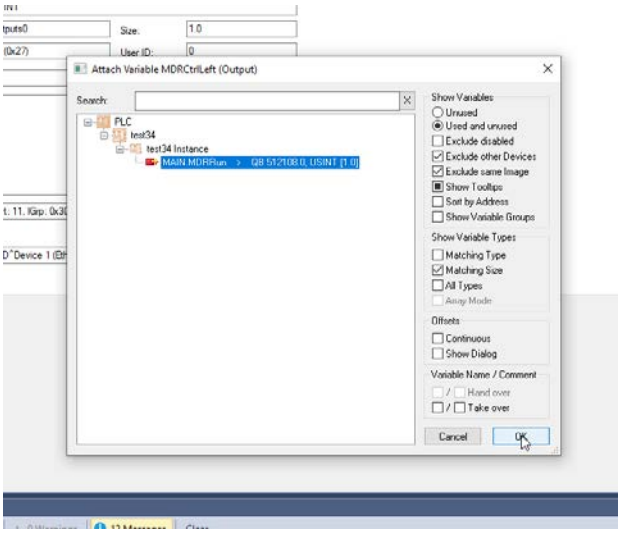


## Assignment of process data in TwinCAT®

1. Open the corresponding process data object of a specific ECDriveS® control module by clicking the arrow on the left side [1].
2. Select the desired process data [2].
3. Double-click to open the menu.
4. In the process data menu, click the "Linked to" button [3].



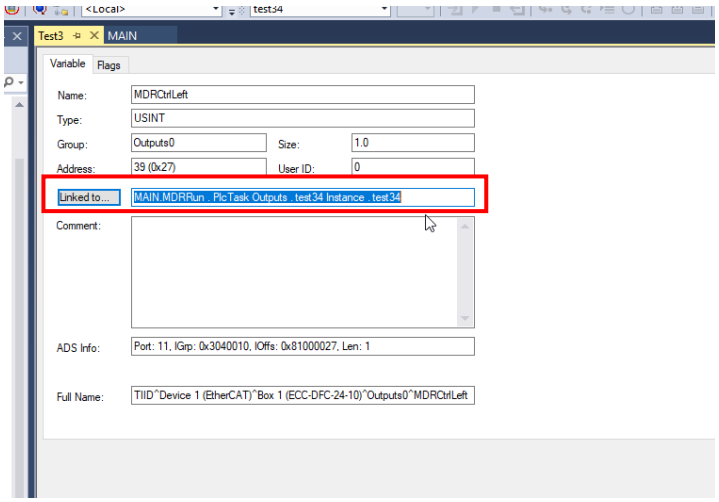

5. In the following menu, you can select an already defined variable of the same data type:




6. Click "OK" to confirm the selection.

INFORMATION

If the selection is successful, the link to the variable will appear next to the "Linked to" function field




### 3.4.5. Configuring additional process data objects

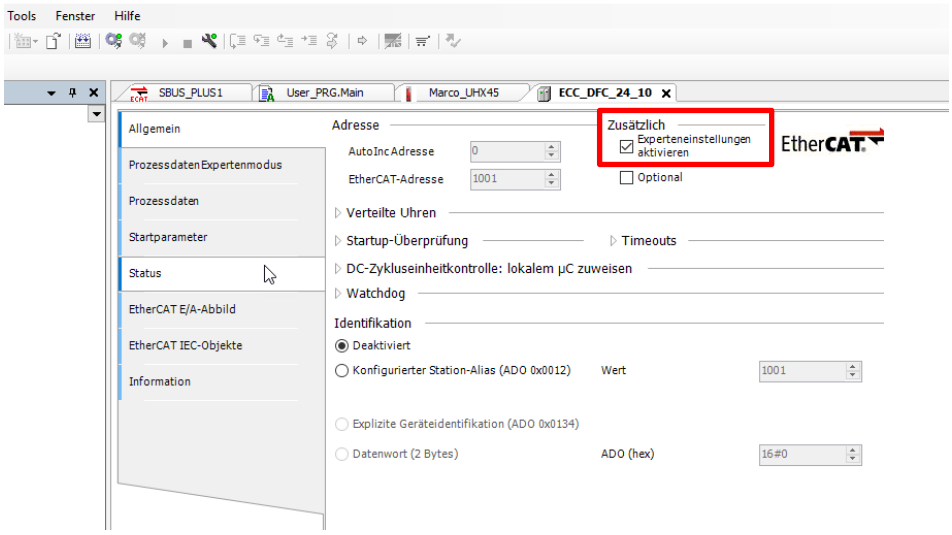
In addition to the standard process data objects from chapter 3.4.4, it is possible to configure a maximum of 3 additional PDOs in each case. The PDOs can be configured with predefined process data of different types and lengths. The process data can be selected from a list. A description of the additional process data can be found in chapter 5.

INFORMATION

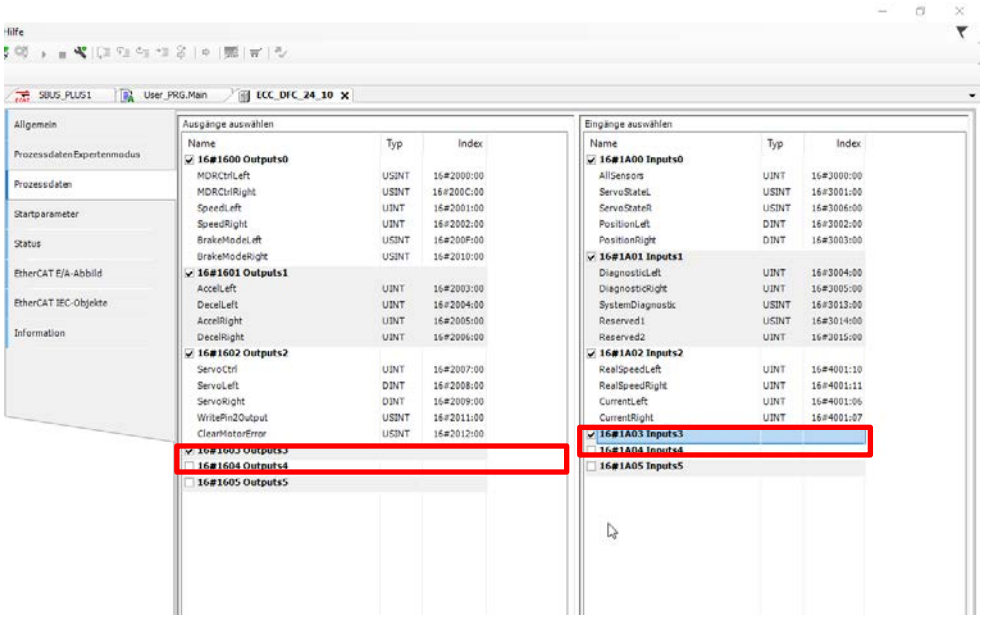
When configuring, make sure that the total data size is a multiple of 4 bytes. In addition, a maximum of 10 process data items can be inserted in a PDO

## Configuration in CODESYS

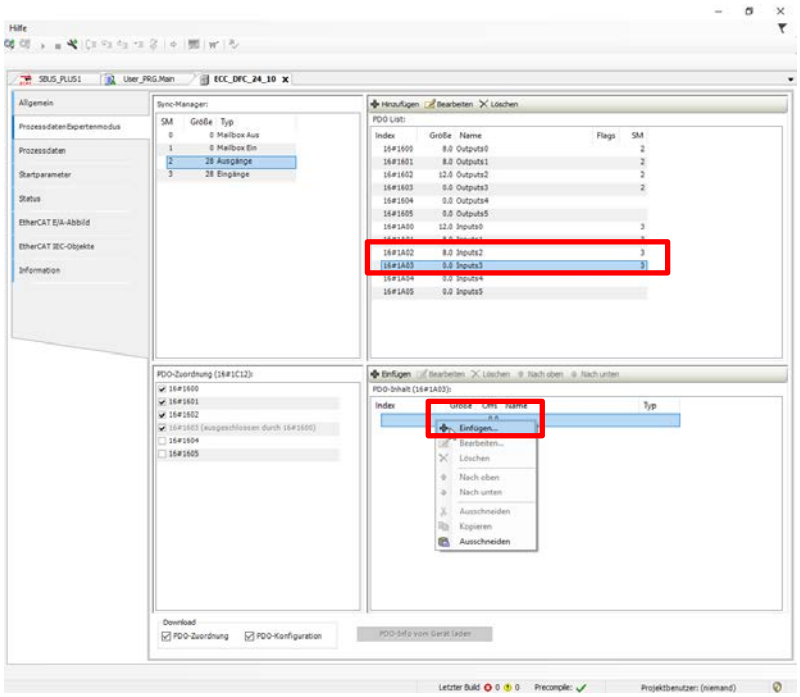
1. Open the menu of the corresponding fieldbus module.
2. In the "General" submenu, the "Activate Expert Settings" field must be activated.



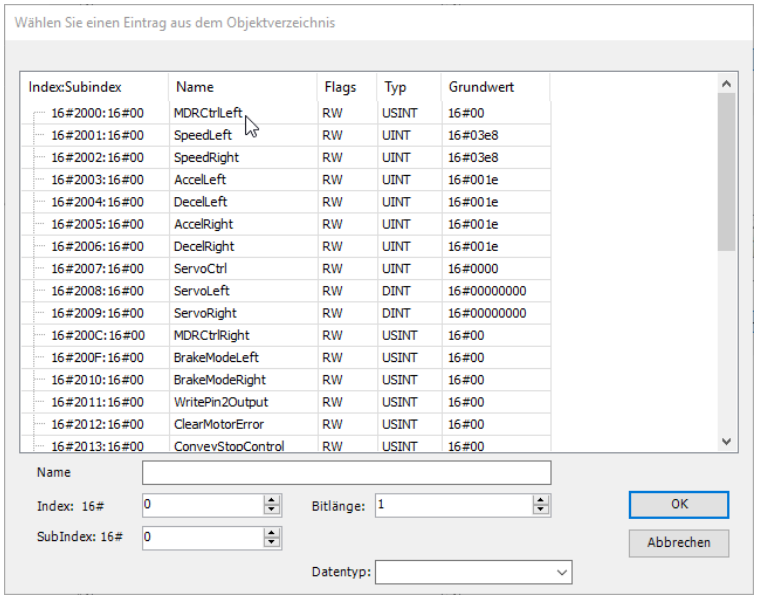

3. In the Process Data submenu, activate the process data objects you need by activating the checkboxes next to the corresponding objects.




4. Then open the "Process Data Expert Mode" submenu. Select the additionally activated process data object in the PDO List and right-click in the PDO Content field. In the shortcut menu that opens, select the "Add" option.

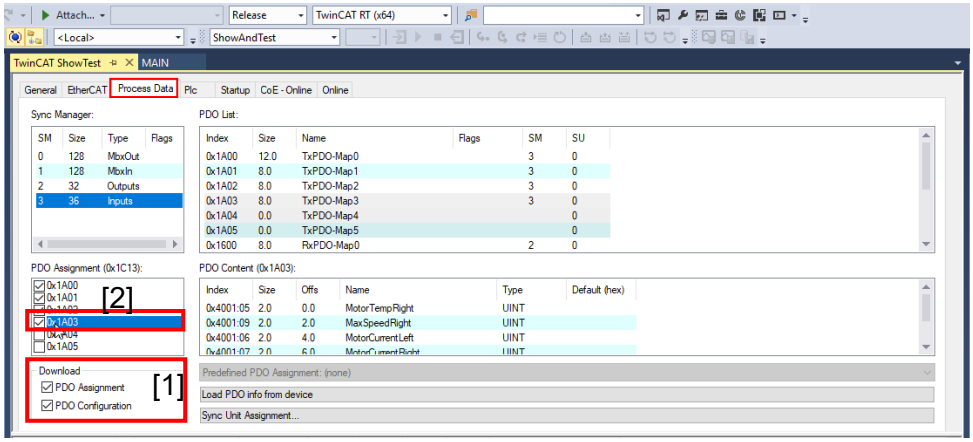



5. In the list that opens, you can configure the process data for the PDO.

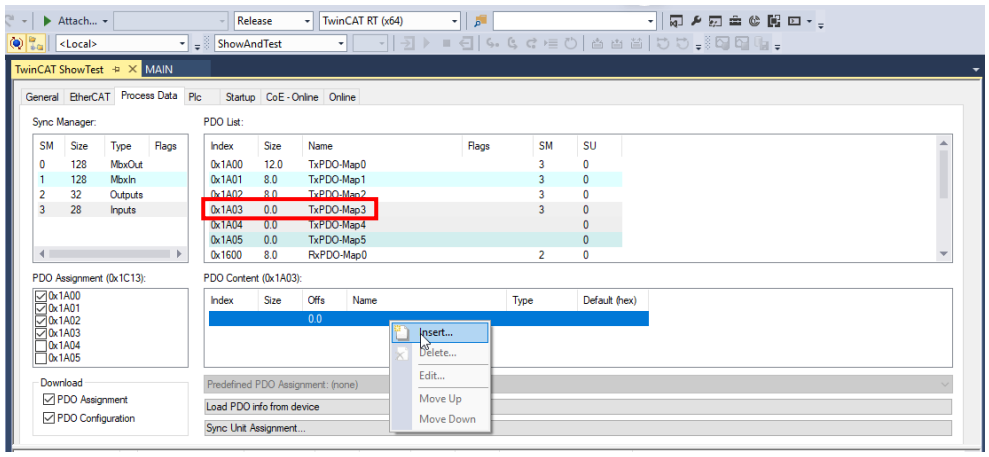



## Configuration in Beckhoff TwinCAT®

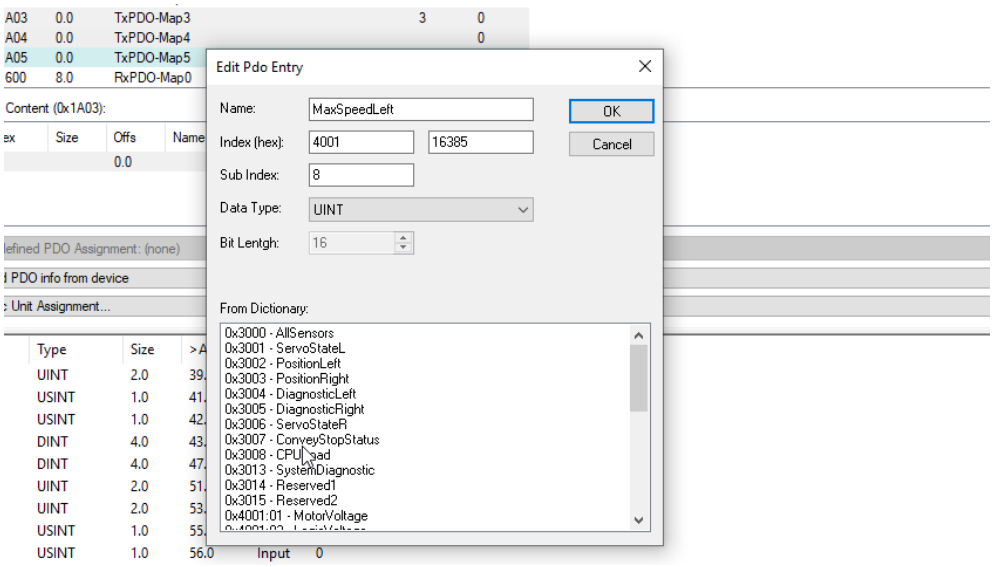
1. Open the menu of the corresponding fieldbus module.
2. Activate the two "PDO Assignment" and "PDO Configuration" fields [1].
3. Activate the PDO you want to add [2].



4. Select the added process data object in the PDO List. Then right-click in the PDO Content window and select the "Insert" function from the shortcut menu.




5. Select the desired process data and confirm the selection by clicking "OK".



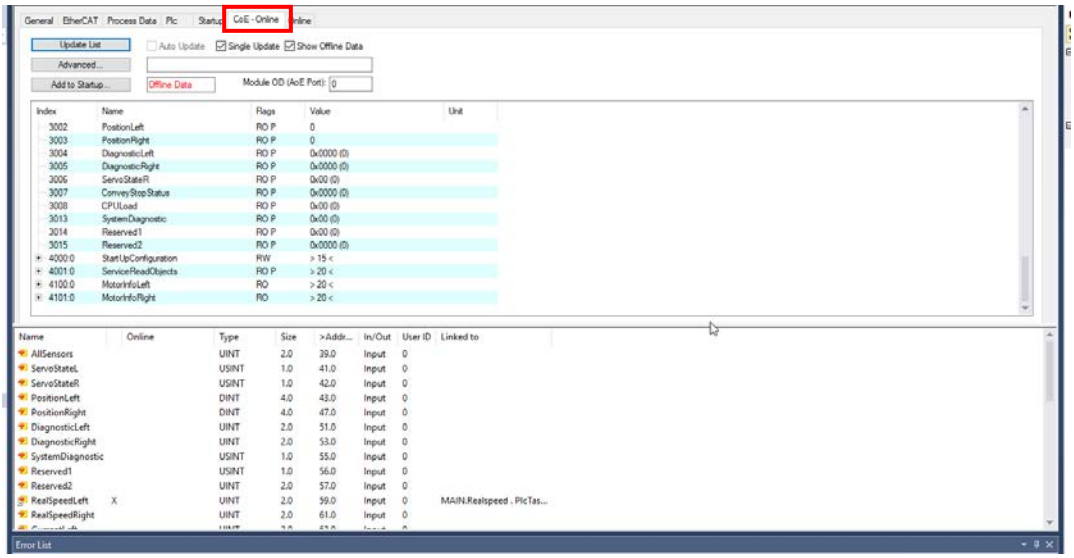

## 3.5. Parameterization

The ECC-DFC-24-10 fieldbus module does not have an integrated parameter structure. Parameterization is performed exclusively via the service data objects (SDOs) in the bus master. The parameter values are therefore stored in the SDO structure of the bus master. A description of the individual SDOs can be found in chapter 3.5.3

### 3.5.1. Access to the SDOs of the ECC-DFC-24-00 module in TwinCAT®

The individual SDOs can be accessed via the CoE service. To open the CoE data, proceed as follows:

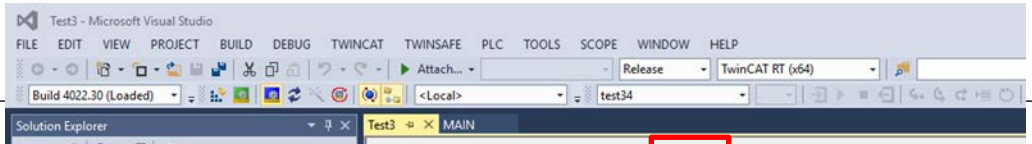
1. Select the main menu of the respective fieldbus module by double-clicking the corresponding module.
2. Select the "CoE - Online" tab.

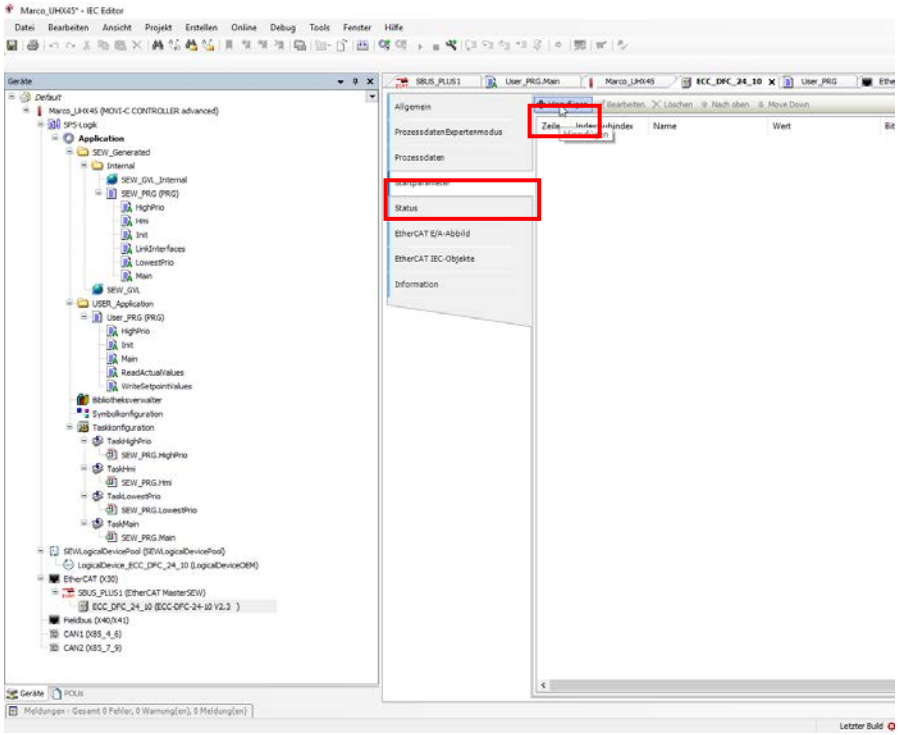



### 3.5.2. Access to the SDOs of the ECC-DFC-24-10 module in CODESYS

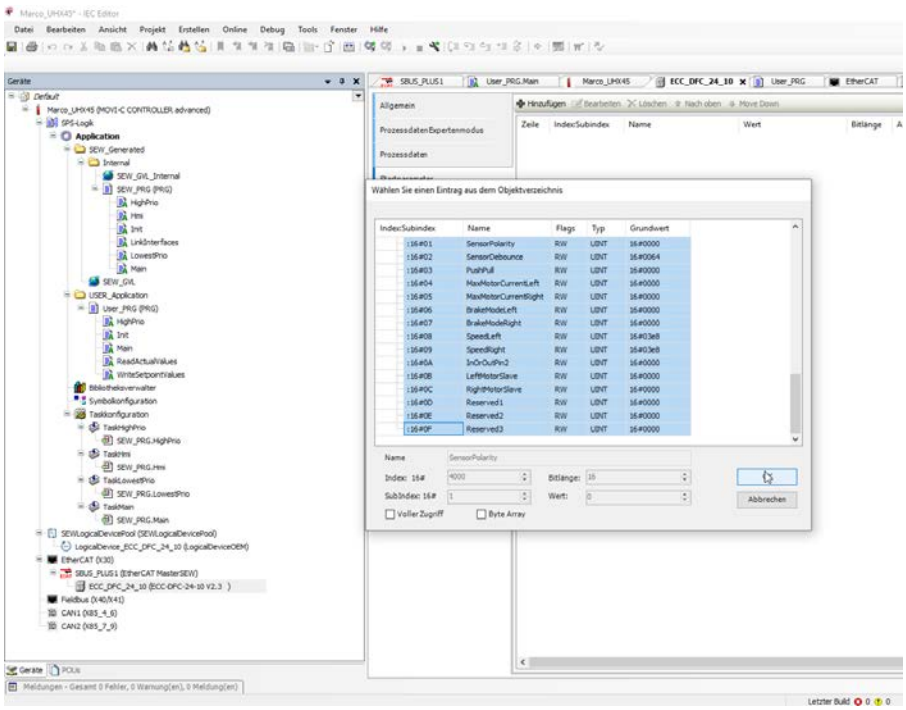
In CODESYS, the service data objects must be integrated into the start parameters.

1. To do this, open the menu by double-clicking the device.
2. Select the "Start Parameters" submenu.
3. Then click the "Add" field.





4. In the list, select the corresponding SDOs as described in section 3.5.3 and confirm your entry by clicking "OK".



### 3.5.3. Description of the service data objects

Index 0x4000 Startup Configuration

| Subindex | Parameter | Data type | Description |
|---|---|---|---|
| 01 | Sensor Polarity | INT | Writing values bit-by-bit<br>Bit 00 = Left_Pin2<br>**True:** Invert sensor input **False:** Default polarity<br>Bit 01 = Reserved<br>Bit 02 = Right Pin2<br>**True:** Invert sensor input **False:** Default polarity<br>Bit 03 = Reserved<br>Bit 04 = Left Pin4<br>**True:** Invert sensor input **False:** Default polarity<br>Bit 05 = Reserved<br>Bit 06 = Right Pin4<br>**True:** Invert sensor input **False:** Default polarity<br>Bit 07 - Bit 15 = Reserved |
| 02 | Sensor Debounce | INT | Time for sensor debouncing in [ms] |
| 03 | Push-Pull Sensor Type | INT | Writing values bit-by-bit<br>Bit 00 = Left Sensor Port Inputs<br>**True:** Sensor type = Push-Pull<br>**False:** Sensor type = NPN/PNP (default)<br>Bit 01 = Right Sensor Port Inputs<br>**True:** Sensor type = Push-Pull<br>**False:** Sensor type = NPN/PNP (default)<br>Bit 02 bis Bit 15 = Reserved |
| 04 | MaxMotorCurrentLeft | INT | Current limit of the left drive<br>**Value 0x0** = 3A (default)<br>**Value 0x1** = 5A<br>**Value 0x2** = 7A |
| 05 | MaxMotorCurrentRight | INT | Current limit of the left drive<br>**Value 0x0** = 3A (default)<br>**Value 0x1** = 5A<br>**Value 0x2** = 7A |
| 06 | BrakeModeLeft | INT | Deceleration behavior of the left drive<br>**Value 0x1:** Standard brake ramp<br>**Value 0x2:** Coasting<br>**Value 0x3:** ServoBrake |

| Subindex | Parameter | Data type | Description |
|---|---|---|---|
| 07 | BrakeModeRight | INT | Deceleration behavior of the right drive<br><br>**Value 0x1:** Standard brake ramp<br>**Value 0x2:** Coasting<br>**Value 0x3:** ServoBrake |
| 08 | SpeedLeft | INT | Speed/rotational speed of the left drive<br><br>**With ECR:** v(ECR) [m/s] x 1000<br>**With ECG:** v(ECG) [inc.][1)] |
| 09 | SpeedRight | INT | Speed/rotational speed of the right drive<br><br>**With ECR:** v(ECR) [m/s] x 1000<br>**With ECG:** v(ECG) [inc.][1)] |
| 10 | InOrOutPin2 | INT | DIO PIN2 function<br>**Value 0x0:** PIN2 = DI (default)<br>**Value 0x1:** Left PIN2 = DO<br>**Value 0x2:** Right PIN2 = DO<br>**Value 0x3:** Both PIN2 = DO |
| 11 | LeftMotorSlave | INT | Slave function left motor<br>**Value 0x0:** Slave function deactivated (default)<br>**Value 0x1:** Left motor follows the control signals of the right motor |
| 12 | RightMotorSlave | | Slave function left motor<br>**Value 0x0:** Slave function deactivated (default)<br>**Value 0x1:** Right motor follows the control signals of the left motor |
| 13 | Reserved | | |
| 14 | Reserved | | |
| 15 | Reserved | | |

## Index 0x4001 - ServiceReadObjects

| Subindex | Parameter | Data type | Description |
|---|---|---|---|
| 01 | MotorVoltage | INT | Motor voltage in [mV] |
| 02 | LogicVoltage | INT | Logic voltage in [mV] |
| 03 | SensorDetect | INT | **Value 0x01: LeftSensorDetect**. A sensor was detected on the left sensor port (PIN4)<br>**Value 0x02**: **RightSensorDetect**: A sensor was detected on the right sensor port (PIN4)<br>**Value 0x03**: A sensor was detected on both sensor ports (PIN4) |
| 04 | MotorTempLeft | INT | Temperature of the left drive train<br>**Low byte**: Motor temperature in [°C]<br>**High byte**: Temperature of the power section of the left drive axis in [°C] |
| 05 | MotorTempRight | INT | Temperature of the right drive train<br>**Low byte**: Motor temperature in [°C]<br>**High byte**: Temperature of the power section of the left drive axis in [°C] |
| 06 | MotorCurrentLeft | INT | Left motor roller current in mA: 5000 for 5A. This value is averaged over a 500 ms interval |
| 07 | MotorCurrentRight | INT | Right motor roller current in mA: 5000 for 5A. This value is averaged over a 500 ms interval |
| 08 | MaxSpeedLeft | INT | Maximum speed / rotational speed of the left motor from electronic nameplate.<br>With ECR: v(ECR) [m/s] x 1000<br>With ECG: v(ECG) [$min^{-1}$] x 10 |
| 09 | MaxSpeedRight | INT | Maximum speed / rotational speed of the right motor from electronic nameplate.<br>With ECR: v(ECR) [m/s] x 1000<br>With ECG: v(ECG) [$min^{-1}$] x 10 |
| 10 | RealSpeedLeft | INT | Measured speed / rotational speed of the left drive.<br>With ECR: v(ECR) [m/s] x 1000<br>With ECG: v(ECG) [$min^{-1}$] x 10 |

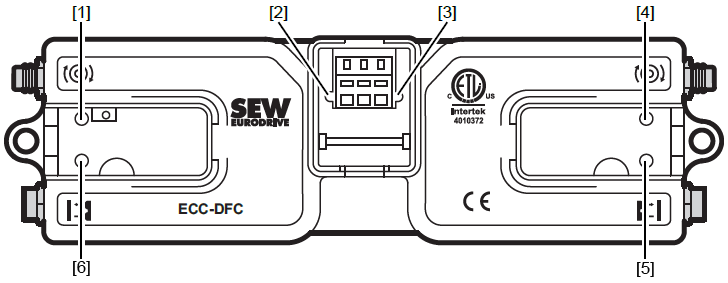
| Subindex | Parameter | Data type | Description |
|---|---|---|---|
| 11 | RealSpeedRight | INT | Measured speed / rotational speed of the right drive.<br>With ECR: v(ECR) [m/s] x 1000<br>With ECG: v(ECG) [$min^{-1}$] x 10 |
| 12 | MDRFrequencyLeft | INT | Current motor frequency in [Hz] of the left drive |
| 13 | MDRFrequencyRight | INT | Current motor frequency in [Hz] of the right drive |
| 14 | MotorPWMLeft | INT | Current PWM with which the left motor is controlled<br>Value in [%] x 10 |
| 15 | MotorPWMRight | INT | Current PWM with which the right motor is controlled<br>Value in [%] x 10 |

## Index 4100 / 4101 - MotorInfoLeft / Right

| Subindex | Parameter | Data type | Description |
|---|---|---|---|
| 01 | Product ID | INT | Production ID |
| 02 | Customer ID | SINT | Customer ID |
| 03 | RollerType | SINT | Roller type, ASCII value<br>A = IP54 degree of protection (standard)<br>W = IP66 degree of protection (wet area)<br>Z = cold storage |
| 04 | MotorType | SINT | Motor type<br>**Value 0x0**: ECR<br>**Value 0x1**: ECG |
| 05 | DiamCode | SINT | Roller diameter (only valid for ECR)<br>Value in [mm] |
| 06 | GearBox | SINT | Gear unit ratio<br>ECR: Value is Speedcode<br>ECG: Value is gear ratio i |
| 07 | Interlock | SINT | Variant overdrive<br>ASCII value<br>e.g. V1 for Poly-V |
| 08 | Month | SINT | Production date: Month |
| 09 | Year | SINT | Production date: Year |
| 10 | Shaft | SINT | Internal SEW-EURODRIVE value |
| 11 | Tube | SINT | Pipe option (from type code)<br>e.g. Z = galvanized steel |
| 12 | MtrLen | INT | Roller length in [mm] |
| 13 | Day | SINT | Production date: Day |
| 14 | Assembled | | Internal SEW-EURODRIVE value |
| 15 | Time | | Runtime of the motor [min] |
| 16 | TimeCurrLimit | | Time at current limit [min] |
| 17 | TimeOverHeat | | Time in overtemperature fault [min] |
| 18 | OnOffCycles | | On/off cycles counter |
| 19 | OverVoltageCounter | | Overvoltage cut-off counter |
| 20 | UnderVoltageCounter | | Undervoltage cut-off counter |

# 4. Operation

## 4.1. Status displays




[1] Motor LED. Left drive

[2] Power LED

[3] Status LED

[4] Motor LED, right drive

[5] Sensor LED, right drive

[6] Sensor LED, left drive

### 4.1.1. Status LED

| Display | Position | LED status | Description |
|---|---|---|---|
| Fieldbus controller status | [3] | Off | EtherCAT in INIT status |
| | | Flashing (2.5 Hz), green | EtherCAT® in PRE-OP status |
| | | Flashing (1 Hz), green | EtherCAT® in SAFE-OP status |
| | | Lights up, green | EtherCAT® in OPERATIONAL status |
| | | Flashing (10 Hz), green | EtherCAT® in BOOTSTRAP status |
| | | Flashing (2.5 Hz), red | Configuration error |
| | | Flashing (1 Hz), red | Local error |
| | | Lights up, red | Critical communication or application error |
| | | Flashing (10 Hz), red | Boot error |

### 4.1.2. Motor LEDs

| Display | Position | LED status | Description |
|---|---|---|---|
| Left motor and right motor | [1] and [4] | Off | Drive is not enabled |
| | | Lights up green | Drive is enabled |
| | | Lights up red | Drive enabled:<br>• Current limiting active<br>Drive not enabled:<br>• Motor connection faulty<br>• Voltage supply outside the nominal range |
| | | Slow flashing | Motor temperature > 107 °C, current limiting active |
| | | Quickly flashing red | Motor short circuit detected between at least 2 of the phase windings |

### 4.1.3. Sensor LEDs

| Display | Position | LED status | Description |
|---|---|---|---|
| Sensors | [5] and [6] | Lights up yellow | Fieldbus controller starts |
| | | Lights up green | Sensor input activated |
| | | Lights up red | PIN2 function activated |
| | | Quickly flashing red | Network stop status |

### 4.1.4. Power LED

| Display | Position | LED status | Description |
|---|---|---|---|
| Supply voltage | [2] | Lights up blue | Supply voltage is present |
| | | Slowly flashing blue | Supply voltage is less than 18 V |
| | | Off | Supply voltage is not present |

Ethernet LEDs

| Display | Position | LED status | Description |
|---|---|---|---|
| Ethernet connection status | [6] and [7] | Lights up green | Ethernet connection available |
| | | Off | No Ethernet connection available |
| | | Quickly flashing yellow | Data transfer via Ethernet in progress |

## 4.2. EtherCAT® Shell

The EtherCAT® Shell engineering and diagnostics software enables access to the ECC-DFC-24-10 fieldbus module independently of the fieldbus master. The software offers the following range of functions:

- Diagnostics by means of virtual status LEDs
- Access to the electronic nameplate
- Fieldbus monitor
- Manual mode
- Physical localization of the device

A distinction is made between two operating modes for accessing the fieldbus module:

- Online mode
  Access to the module during active fieldbus operation (restricted range of functions)
- Offline mode
  Access to the module when the master is not active

### 4.2.1. Online mode

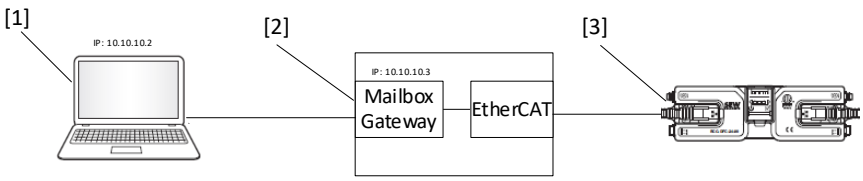
In online mode, the following functions are available:

- Fieldbus monitor
- Virtual status LEDs

- Electronic nameplate

To use the online mode, you must configure a mailbox gateway connection in your fieldbus master (for further information about this, please refer to the description by the corresponding manufacturer).

### Example of network topology



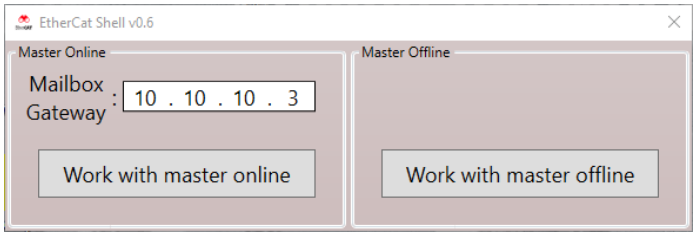

[1] Engineering PC

[2] EtherCAT master

[3] ECC-DFC-24-10

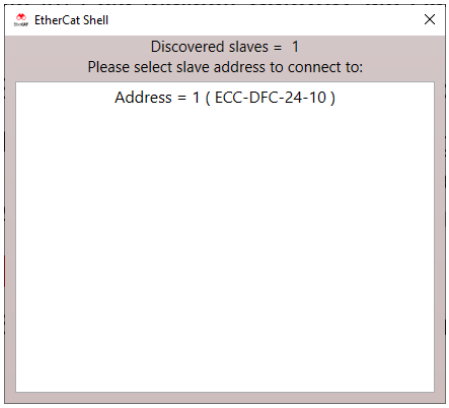
### Opening the online mode

After the software has been opened, the following start screen is displayed:




1. In the "Mailbox Gateway" field, enter the address of your configured mailbox gateway.
2. Then click the "Work with master online" function.

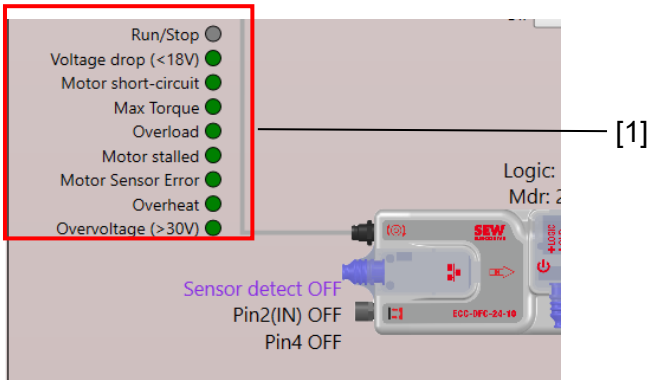
After the mailbox gateway has been set up successfully in your fieldbus master and the associated configuration of the network structure has been carried out, a selection field appears listing the ECC-DFC-24-10 stations found.




3. Select the module that is relevant for you and confirm the selection by double-clicking it.

### 4.2.2. EtherCAT Shell status display

The EtherCAT Shell status display shows the current motor state with the help of virtual status LEDs. These real-time displays are shown separately for each motor. A description of the errors and warnings can be found in the operating instructions "ECDriveS® ECC-DFC fieldbus controller, ECR roller drives, ECG gearmotor".



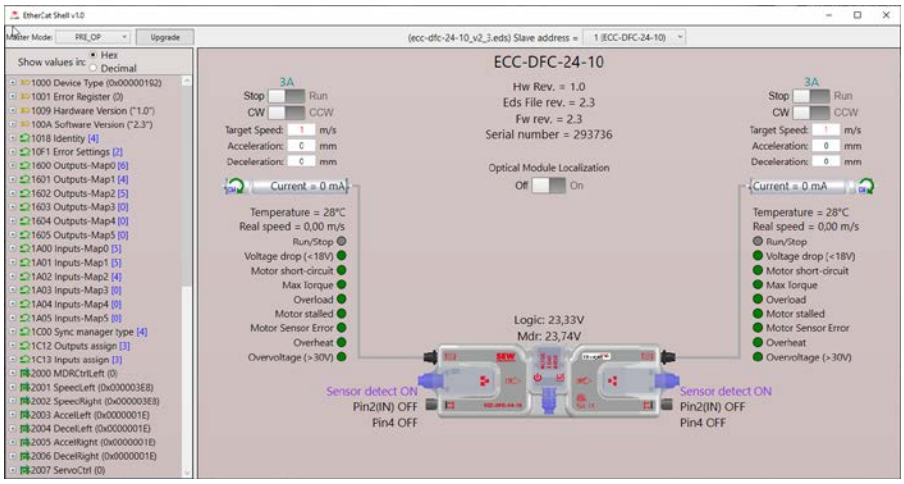

[1] Virtual status LEDs

The virtual status LEDs have 3 possible states:

- Green: Normal operation
- Yellow: Warning
- Red: Fault status

### 4.2.3. Graphic module status display

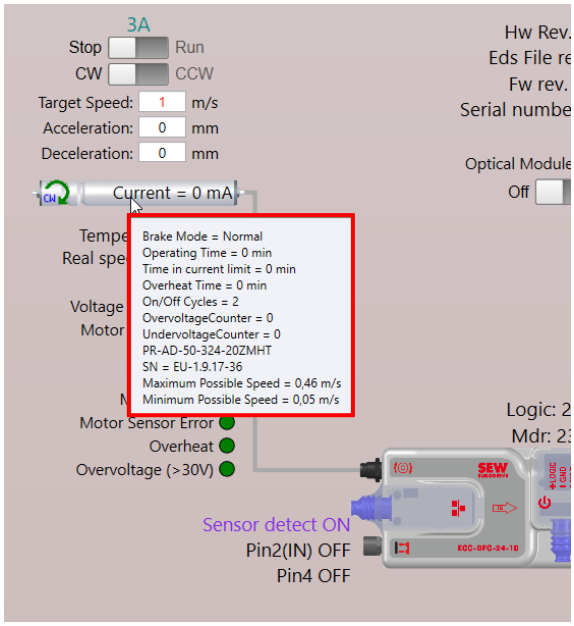
The graphic display of the module state of EtherCAT Shell allows for a virtual real-time display of the selected ECC-DFC fieldbus controller. The display provides information about the connected periphery. Further, the electronic nameplate of the motor can be displayed.




### 4.2.4. Electronic nameplate

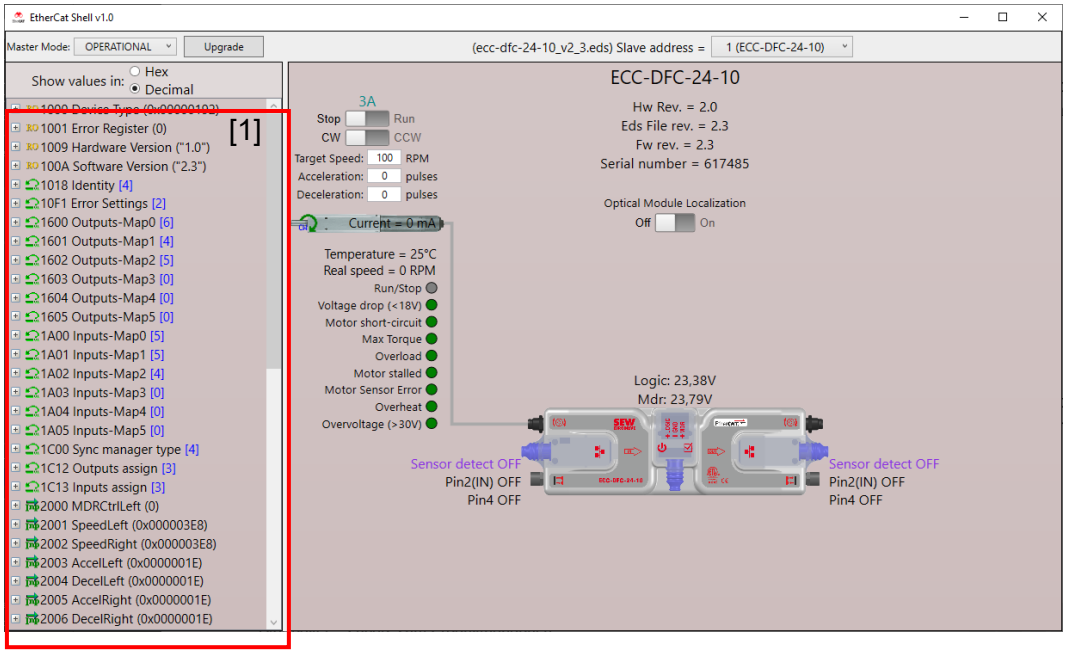
If you move the mouse pointer over a connected motor, a dialog field opens with further information about current parameter settings as well as status and error displays.

The parameter settings are described in the chapters "Parameterization and Service". In addition, the type of the connected motor and its speed range (maximum speed, minimum speed) are displayed here.




### 4.2.5. Fieldbus monitor

The fieldbus monitor is a real-time display of incoming and outgoing process data. This makes it possible to check the data traffic between the ECC-DFC and the fieldbus master independently of the user program.




[1] Fieldbus monitor

Identifying symbols

| Symbol | Designation | Description |
|---|---|---|
|  | ReadOnlyRegister | Read access only |
|  | ReadWriteRegister | Register/index can be read and written |
|  | Object view | **Process Data Object**<br>This shows the structure of the process data object and is used to identify the process data to be expected<br><br>Example:<br>Object 1600<br>r/w sub1 Subindex1 (0x20000008)<br>Process data of the index 0x2000.00 with a length of 8 bits should be available at this position<br><br>**Service Data Object (SDO)**<br>This shows the structure and content of the corresponding SDO |
|  | Process Data Input (RX) | Incoming process data from the fieldbus module's point of view |
|  | Process Data Output (TX) | Outgoing process data from the fieldbus module's point of view<br><br>Values that are used as process and service data (e.g. motor current) are located in the service data object (0x4000; 0x4001) |

### 4.2.6. Optical module localization

You can localize a specific fieldbus controller in a system by using optical module localization. The "Optical module localization" function can be accessed via the EtherCAT Shell main menu.

Activating the function

1. To activate the function, click the "On" button.

⇨ The motor LED and the sensor LEDs of the selected fieldbus controller flash red.

Deactivating the function

The "Optical module localization" function must be reset manually.

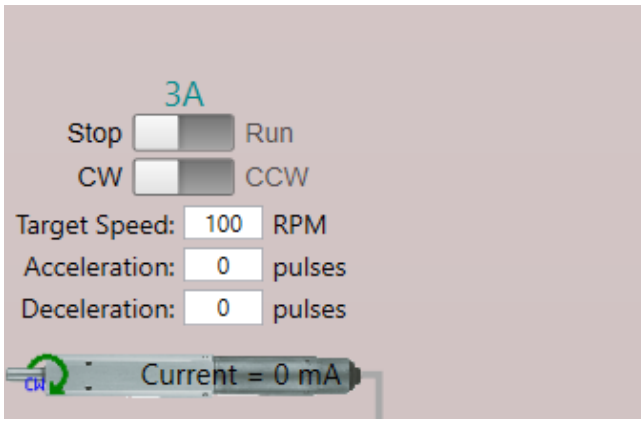
2. To deactivate the function, click the "Off" button.

⇨ The motor LED and the sensor LEDs of the selected fieldbus controller no longer flash red.

### 4.2.7. Manual mode

The EtherCAT Shell engineering and diagnostics software offers the option of manual operation. This means it is possible to make the connected motors rotate independently of the control. This is necessary, for example, during system startup when the external control is not yet in operation and a machine movement is required. The manual mode of the ECC-DFC-24-10 fieldbus module offers the following range of functions:

- Enable
- Change of the direction of rotation
- Changing the acceleration and deceleration ramps
- Changing the speed/rotational speed




INFORMATION

Manual mode is only available in the offline mode

# 5. Appendix

## 5.1. Directory of process input data objects

### 5.1.1. Standard process data objects

0x1A00 - Inputs0

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x3000.0 | All Sensors | INT | Reading values bit-by-bit<br>Bit 00: **LeftPIN2** = status of the left sensor port (PIN2)<br>Bit 01: Reserved<br>Bit 02: **RightPIN2** = status of the right sensor port (PIN2)<br>Bit 03: Reserved<br>Bit 04: **LeftPIN4** = status of the left sensor port (PIN4)<br>Bit 05: Reserved<br>Bit 06: **RightPIN4** = status of the right sensor port (PIN4)<br>Bit 07 to bit 14 = reserved<br>Bit 15 = **Heartbeat** |
| 0x3001.0 | ServoStateL | USINT | Status word for servo commands of the left motor.<br>**Value 0x02:** Referencing completed,<br>DistanceLeft = 0<br>**Value 0x05:** Target position was reached,<br>DistanceLeft = setpoint position |
| 0x3006.0 | ServoStateR | USINT | Status word for servo commands of the right motor.<br>**Value 0x02:** Referencing completed,<br>DistanceLeft = 0<br>**Value 0x05:** Target position was reached,<br>DistanceLeft = setpoint position |
| 0x3002.0 | PositionLeft | DINT | Current motor position S1 of the left motor<br>For ECR: S1(ECR) [mm]<br>For ECG: S1(ECG) [inc.] |
| 0x3003.0 | PositionRight | DINT | Current motor position S1 of the right motor<br>For ECR: S1(ECR) [mm]<br>For ECG: S1(ECG) [inc.] |

## 0x1A01 - Inputs1

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x3004.0 | DignosticLeft | INT | Reading values bit-by-bit<br>Bit 00: **MDRStatus Bit1** = description of the 2-bit coded statuses<br>Bit 01: **MDRStatus Bit2** = description of the 2-bit coded statuses<br>Bit 02 - bit 04 = reserved<br>Bit 05: **CPUOverheat** = overtemperature fault. The heat sink temperature of the fieldbus controller is > 91 °C<br>Bit 06: **OverVoltage** = overvoltage fault<br>Bit 07: **LowVoltage** = undervoltage fault<br>Bit 08: **Overheat** = overtemperature fault. The temperature of the motor roller has exceeded 105 °C<br>Bit 09: **MaxTorque** = current limit reached<br>Bit 10: **ShortCircuit**: Motor short circuit<br>Bit 11: **MDRNotConnected** = no motor is connected to the motor port<br>Bit 12: **Overload** = overload detected. Speed lowered to <10% of the setpoint speed due to the overload.<br>Bit 13: **Stalled** = motor blockage detected<br>Bit 14: **MDRBadHall** = Hall sensor fault<br>Bit 15 = **MDRNotConnected:** No motor is connected to the motor port |
| 0x3005.0 | DignosticRight | INT | |
| 0x3013.0 | SystemDiagnostic | UINT | Reading bit-by-bit<br>Bit 0x00: Logic circuit voltage < 14 V<br>Bit 0x01: Speed setpoint > maximum of the left motor[1]<br>Bit 0x02: Speed setpoint < minimum of the left motor[1]<br>Bit 0x03: Speed setpoint > maximum of the right motor[1]<br>Bit 0x04: Speed setpoint < minimum of the right motor[1] |

## 0x1A02 - Inputs2

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x4001.0A | RealSpeedLeft | UINT | Measured speed/rotational speed of the drive<br>with ECR: v(ECR) [m/s] x 1000<br>with ECG: v(ECG) [$min^{-1}$] x 10 |
| 0x4001.0B | RealSpeedRight | UINT | |

[1] If the setpoint is larger or smaller than the limits of the connected motor, the system automatically adapts the speed/rotational speed to the limits

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x4001.06 | CurrentLeft | UINT | Left motor roller current in mA: 5000 for 5A. This value is averaged over a 500 ms interval |
| 0x4000.07 | CurrentRight | UINT | Right motor roller current in mA: 5000 for 5A. This value is averaged over a 500 ms interval |
| 0x3003.0 | PositionRight | DINT | Current motor position S1 of the left motor<br>For ECR: S1(ECR) [mm]<br>For ECG: S1(ECG) [inc.] |

### 5.1.2. Additional process data

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x3007.0 | ConveyStopStatus | UINT | Status of the Convey Stop function, reading bit-by-bit<br>**Value 0x00**: Normal operation<br>**Value 0x02**: ConveyStop activation by PLC |
| 0x3008.0 | CPULoad | UINT | CPU capacity utilization of main processor in [%] |

INFORMATION

It is also possible to generate some process data from the object 0x4001.
A description of object 0x4001 can be found in chapter 3.5.3

## 5.2. Directory of process output data objects

### 5.2.1. Standard process data objects

0x1600 - Outputs0

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x2000.0 | MDRCtrLeft | USINT | Writing values bit-by-bit |

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x200C.0 | MDRCtrRight | USINT | Bit00: **RUN_MDR** = enable motor<br>**True:** Enable<br>**False:** Stop<br>Bit01: **MDR_Direction** = changeover of direction of rotation<br>**True:** Direction of rotation not equal to preset direction of rotation<br>**False:** Default direction of rotation |
| 0x2001.0 | SpeedLeft | UINT | Speed/rotational speed of the drive<br>with ECR: v(ECR) [m/s] x 1000<br>with ECG: v(ECG) [$min^{-1}$] x 10 |
| 0x2002.0 | SpeedRight | UINT | |
| 0x200F.0 | BrakeModeLeft | USINT | Deceleration behavior of the drive<br>**Value 0x00:** Standard brake ramp<br>**Value 0x01:** Coasting<br>**Value 0x02:** ServoBrake |
| 0x2010.0 | BrakeModeRight | USINT | |

## 0x1601 - Outputs1

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x2003.0 | AccelLeft | UINT | Acceleration of the left drive<br>With ECR: a(ECR)[mm]<br>With ECG: a(ECG)[inc.][2] |
| 0x2004.0 | DecelLeft | UINT | Deceleration of the left drive<br>With ECR: a(ECR)[mm]<br>With ECG: a(ECG)[inc.][2] |
| 0x2005.0 | AccelRight | UINT | Acceleration of the right drive<br>With ECR: a(ECR)[mm]<br>With ECG: a(ECG)[inc.][2] |
| 0x2006.0 | DecelRight | UINT | Deceleration of the right drive<br>With ECR: a(ECR)[mm]<br>With ECG: a(ECG)[inc.][2] |

## 0x1602 - Outputs2

[2] pulses [inc.]/motor revolution.

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x2007.0 | ServoCtrl. | UINT | Writing values bit-by-bit<br>Bit 00: **Left Drive Reference** = sets value ServoLeft = 0<br>Bit 01: **Left Drive Positioning =** left drive positions to ServoLeft position<br>Bit 02 - Bit 07: Reserve<br>Bit 08: **Right Drive Reference** = sets value ServoRight = 0<br>Bit 09: **Right Drive Positioning =** right drive positions to ServoRight position<br>Bit 10 - Bit 15: Reserve |
| 0x2008.0 | ServoLeft | DINT | Target position value for the left motor, absolute incremental value, 32 bit |
| 0x2009.0 | ServoRight | DINT | Target position value for the right motor, absolute incremental value, 32 bit |
| 0x2011.0 | WritePin2Output | USINT | Writing values bit-by-bit<br>**Value 0x01:** Control LeftPIN2 output<br>**Value 0x02:** Control RightPIN2 output<br>**Value 0x03:** Control both outputs<br><br>Index 4000.10 (SDO) must be set before the PIN2 output is activated. |
| 0x2012.0 | ClearMotorError | USINT | Error reset<br>**Value 0x01**: Reset[1)] |

### 5.2.2. Additional process data

| Index | Designation | Data type | Description |
|---|---|---|---|
| 0x2013.0 | ConveyStopControl | USINT | Writing values bit-by-bit<br>**Value 0x00:** ConveyStop deactivated<br>**Value > 0x00:** ConveyStop activated |

INFORMATION

It is also possible to generate some process data from the object 0x4000.
A description of object 0x4000 can be found in chapter 3.5.3

### 5.2.3. Description of motor status bits

| Bit 1 | Bit 2 | Description |
|---|---|---|
| 0 | 0 | Motor is at standstill, standard or ServoBrake activated |
| 0 | 1 | Motor turns counterclockwise |
| 1 | 0 | Motor turns clockwise |
| 1 | 1 | Motor is at standstill and brake is deactivated (freewheel) |