

## Quick Start: MiniMACS6 with TwinCAT 3

<b>Date</b>	Januar 16, 2025
<b>Document number</b>	zd000275-01
<b>Document index</b>	002
<b>Specific use</b>	Document for external use
<b>Change History</b>	000 Initial document
	001 Adjustment to the current xml file
	002 Add DS402 CSP Drive Scheme

### Content

1	Information about the document	2
2	Mapping schemes	2
3	Include the ESI device description	3
4	Creating a new Project	4
5	Include Devices	6
6	SDO Read/ Write	9

## 1 Information about the document

This document shows how a motion controller from maxon | zub can be used with a Beckhoff PLC as an EtherCAT slave. A MiniMACS6 is used as the controller in this example. The engineering tool used is TwinCAT 3 from Beckhoff automation AG.

This document is directly related to the device description file, which can be found in the same ZIP folder.

## 2 Mapping schemes

MACS controllers have various application possibilities and can be used as a conventional slave or as a standalone master. For this reason, there are different mapping schemes.



### 2.1 MACS CiA402 Slave Scheme

This scheme is selected by default when adding a device with multiple mapping schemes. In this scheme the controller is added as a drive with four axis. The axis can then be used in a motion project as NC or CNC axis in CSP mode. The link between the device and the motion project is done automatically.

A DS402 application is required, which is loaded onto the motion controller (MiniMACS6). This can be requested directly from mzub and is free of charge. Minimal ApossC knowledge is required.

### 2.2 MACS Master Scheme

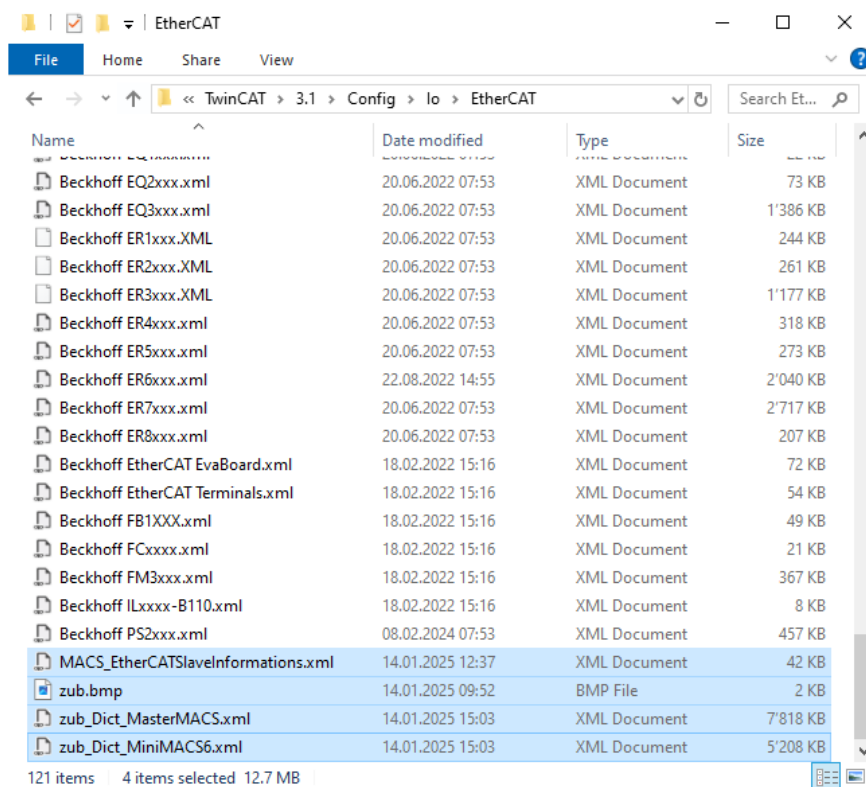
This scheme is used if a user specific exchange between the EtherCAT Master and the MiniMACS6 shall be made. In this case a master application is loaded on the MACS controller.

## 3 Include the ESI device description

In this example, the Beckhoff PLC operates as an EtherCAT master. The master requires the device description file of the motion controller to create the configuration. For the device description ESI (EtherCAT Slave Information) files are used. These are to be stored as \*.xml files in the TwinCAT installation directory. The default installation directory of TwinCAT 3 has the following address:

<C:\TwinCAT\3.1\Config\Io\EtherCAT>

Open the directory of the configuration files.

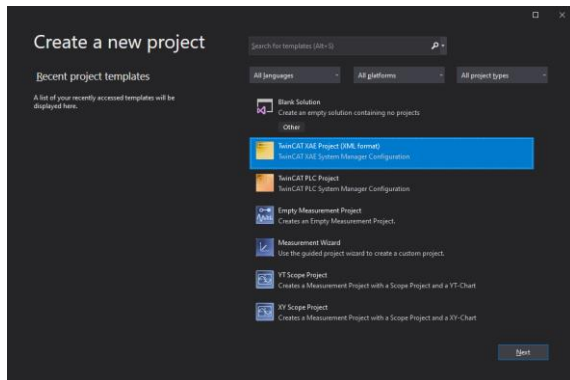


Save the \*.xml file supplied by maxon | zub in this directory. The directory can contain several versions of the same device description.

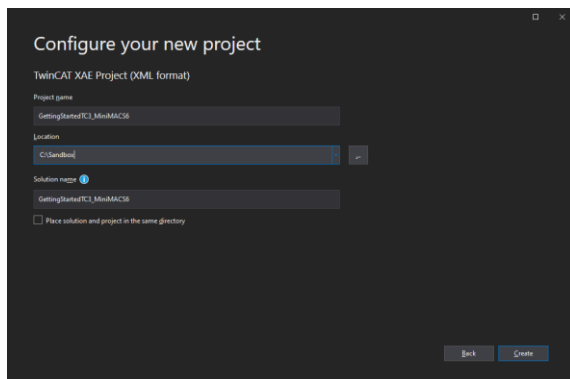
## 4 Creating a new Project

This section shows how a new project can be created with TwinCAT 3. If the controller is included in an existing project, this section can be skipped.

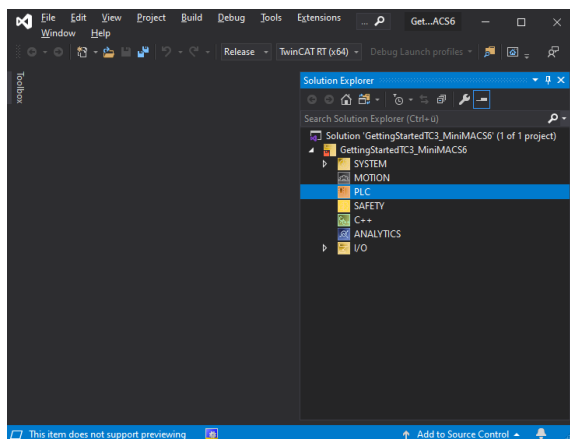
Start TwinCAT3 and create a new project. The TwinCAT XAE Project (XML format) can be used as a project template.



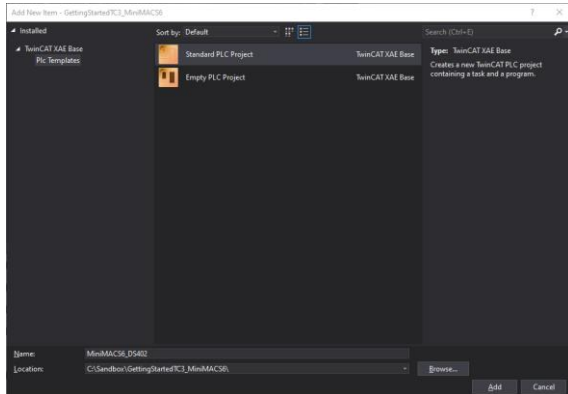
Afterwards the storage path can be specified.



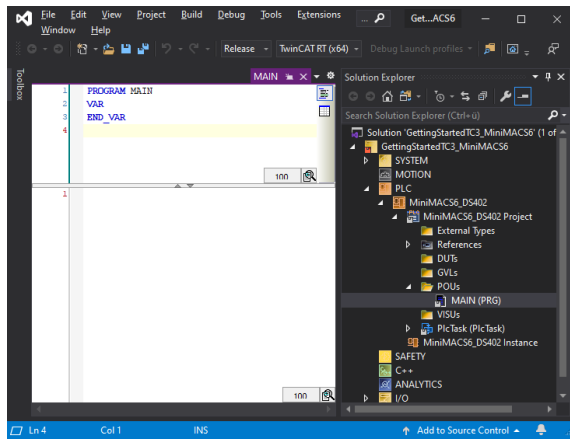
Now a new PLC project can be added. For this add a new item with right click on "PLC".



The template for the project is "Standard PLC Project".



A new project was created. This contains the project with a MAIN routine and project instances.

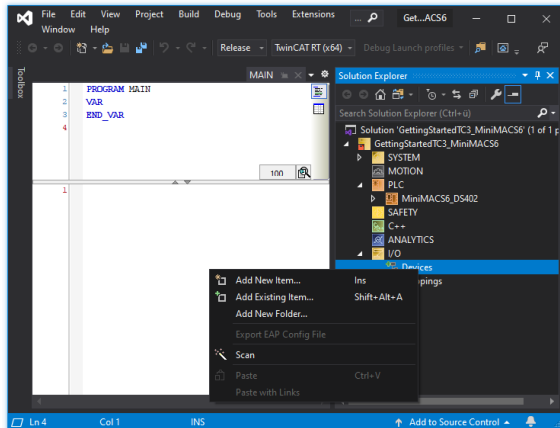


## 5 Include Devices

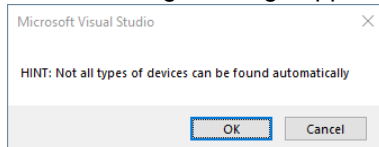
A device can be added automatically by scan or manually. If the **Error! Reference source not found.** or **Error! Reference source not found.** profile is used, the device must be added manually.

### 5.1 Include Devices automatic (MACS CiA402 Slave Scheme)

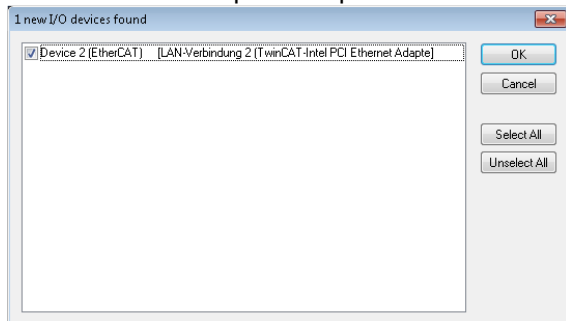
When the device is added with scan, MACS CiA402 Slave Scheme is automatically selected. Double-click on „I/O“. Then right-click on "Devices" and select "Scan".



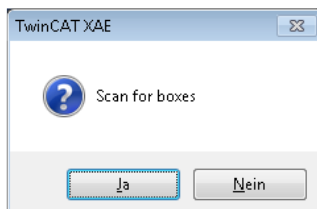
If the following message appears press Ok.



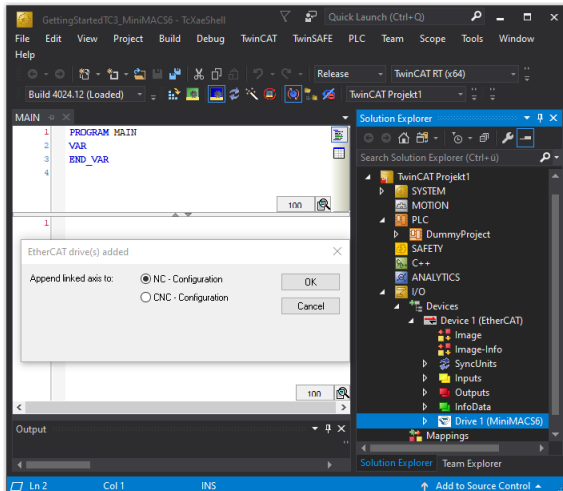
Select network adapter and press OK.



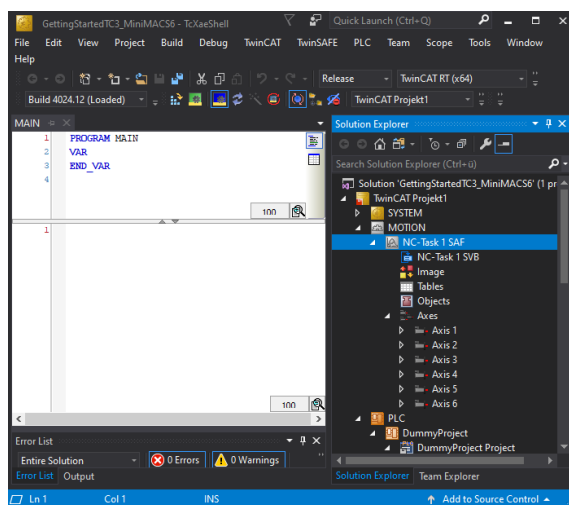
Confirm "scan for boxes" with yes.



The device is added automatically. The request for free run to be activated can be confirmed with yes. Now it can be selected whether the control should be loaded in the NC or CNC configuration. If no NC or CNC motion library is used by, it is possible to cancel the addition of the drive (the device will be added anyway).



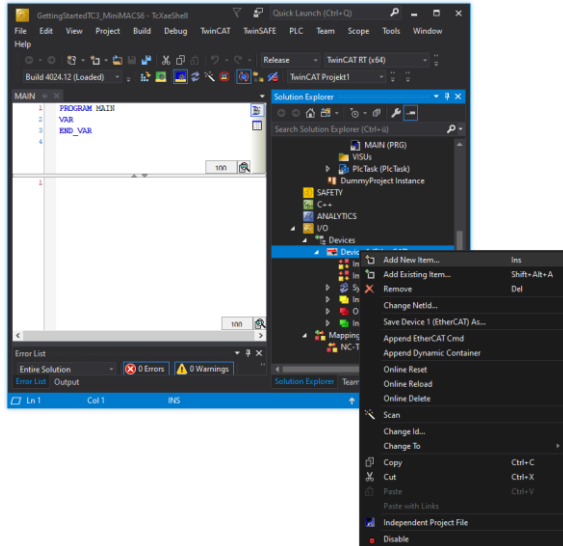
The axis are automatically linked to the respective inputs and outputs.



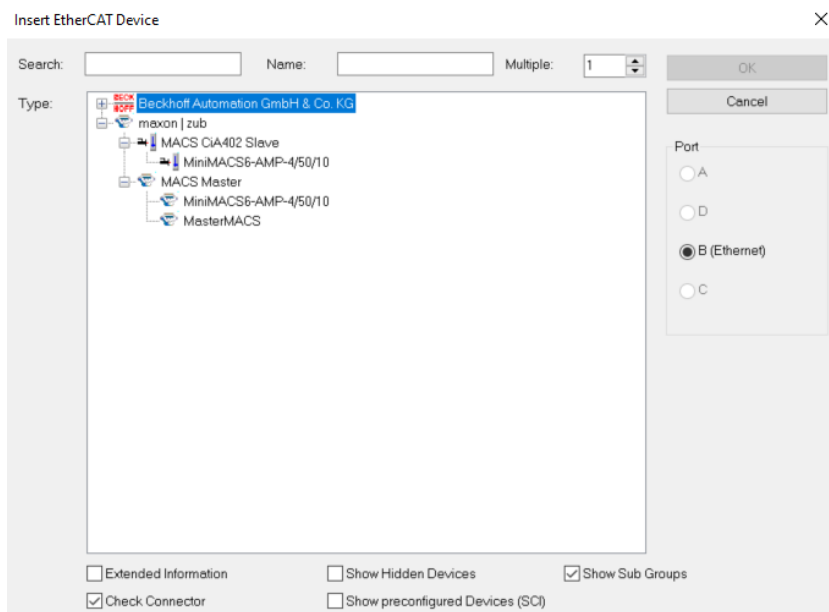
The MiniMACS6 was successfully added.

## 5.2 Include Devices manually

By right-clicking on the correct EtherCAT node, a device can be added manually. For this purpose "Add New Item" must be selected.



Now the user can choose between the three schemes.



The MiniMACS6 has subsequently been added successfully. The linking must be done manually.



## 6 SDO Read/ Write

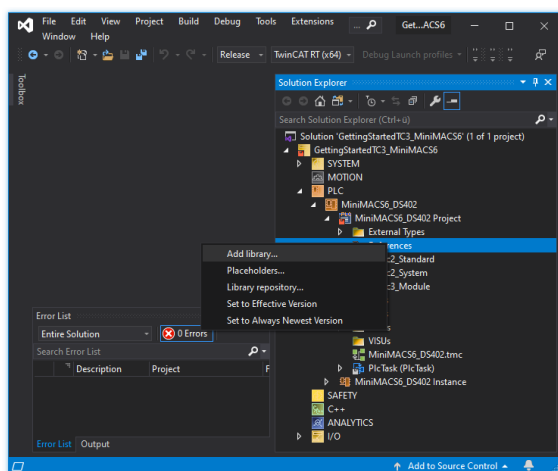
This section shows a brief overview of how the Beckhoff SDO Read/Write functionality can be used.

### 6.1 Include TwinCAT EtherCAT library

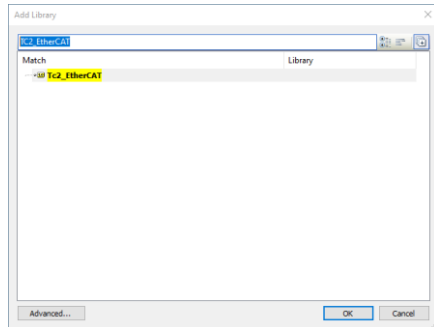
To use the SDO read/ write functionality of TwinCAT the corresponding library must be added. If this library is added, the SDO of the motion controller can be accessed.

[PLC Lib: Tc2 EtherCAT](#)

In the project you can add a library in the references area by right clicking.



Afterwards the term "TC2\_EtherCAT" can be searched in the search field. By selecting the library and confirming the ok button it will be added.



The two function blocks "FB\_EcCoESdoWrite" and "FB\_EcCoESdoRead" are now available. To use these function blocks correctly there are minimal examples on the [Beckhoff info page](#).

## 6.2 Struct AccesSDO

It is recommended to create a structure for read and write access via SDO. The data type is included in the QuickStart package and can be added in an application.

```

TYPE AccessSDO :
STRUCT
    nIndex      : WORD;      (* CoE Object Index *)
    nSubIndex   : BYTE;      (* Subindex of CoE Object *)
    nValue      : DINT;      (* variable to be w/ r to the CoE Object *)
    bExecute    : BOOL;      (* rising edge starts w/ r to the CoE Object *)
END_STRUCT
END_TYPE

```

## 6.3 FB\_SDOWrite

This is a suggestion of what a function block for writing an SDO might look like. The function module is included in the QuickStart package and can be added in an application.

```

FUNCTION_BLOCK FB_SDOWrite
VAR_IN_OUT
    SDOWrite      :AccessSDO;      (* rising edge starts writing to the CoE Object *)
END_VAR
VAR_INPUT
    nSlaveAddr    :UINT;           (* Port Number of EtherCAT Slave *)
    sNetId        :T_AmsNetId;     (* NetId of EtherCAT Master *)
END_VAR
VAR_OUTPUT
    bError        : BOOL;
    nErrId        : UDINT;
END_VAR
VAR
    fbSdoWrite    :FB_EcCoESdoWrite;
END_VAR

fbSdoWrite(      sNetId:=sNetId,
                 nSlaveAddr:=nSlaveAddr,
                 nIndex:=SDOWrite.nIndex,
                 nSubIndex := SDOWrite.nSubIndex,
                 pSrcBuf    := ADR(SDOWrite.nValue),
                 cbBufLen   := SIZEOF(SDOWrite.nValue),
                 bExecute   := SDOWrite.bExecute
                );

IF NOT fbSdoWrite.bBusy THEN
    SDOWrite.bExecute := FALSE;
    IF NOT fbSdoWrite.bError THEN
        (* write successful *)
        bError := FALSE;
        nErrId := 0;
    ELSE
        (* write failed *)
        bError := fbSdoWrite.bError;
        nErrId := fbSdoWrite.nErrId;
    END_IF
    fbSdoWrite(bExecute := FALSE);
END_IF

```

## 6.4 FB\_SDORead

This is a suggestion of what a function block for reading an SDO might look like. The function module is included in the QuickStart package and can be added in an application.

```

FUNCTION_BLOCK FB_SDORead
VAR_IN_OUT
    SDORead      :AccessSDO;      (* rising edge starts reading to the CoE Object *)
END_VAR
VAR_INPUT
    nSlaveAddr   :UINT;           (* Port Number of EtherCAT Slave *)
    sNetId       :T_AmsNetId;     (* NetId of EtherCAT Master *)
END_VAR
VAR_OUTPUT
    bError       : BOOL;
    nErrId       : UDINT;
END_VAR
VAR
    fbSdoRead    :FB_EcCoESdoRead;
END_VAR

fbSdoRead(
    sNetId:= sNetId,
    nSlaveAddr :=nSlaveAddr,
    nIndex:=SDORead.nIndex,
    nSubIndex :=SDORead.nSubIndex,
    pDstBuf:= ADR(SDORead.nValue),
    cbBufLen:=SIZEOF(SDORead.nValue),
    bExecute:=SDORead.bExecute);

IF NOT fbSdoRead.bBusy THEN
    SDORead.bExecute := FALSE;
    IF NOT fbSdoRead.bError THEN
        (* read successful *)
        bError := FALSE;
        nErrId := 0;
    ELSE
        (* read failed *)
        bError := fbSdoRead.bError;
        nErrId := fbSdoRead.nErrId;
    END_IF
    fbSdoRead(bExecute := FALSE);
END_IF

```