User Manual

A000T0075, valid for C-887 controller with EtherCAT interface
FRIE, SSc, BRo, 2/24/2020

Implementing a C-887 PI Controller in TwinCAT 3.1 for Motion and Activation of new Coordinate Systems

For more information on TwinCAT MC2 libraries refer to the Beckhoff Information System (http://infosys.beckhoff.com/content/1033/tcplclib_tc2_mc2/index.html?id=4786081293094367280).

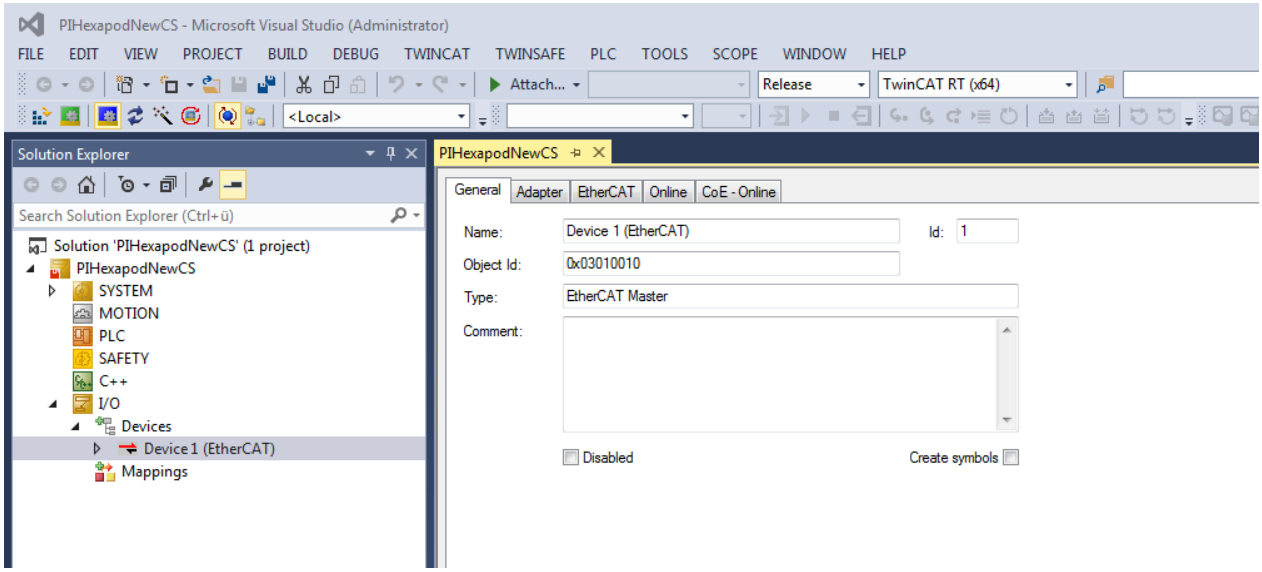
The example described in this document can be found in the \Samples\EtherCAT\TwinCAT directory of the PI software CD delivered with your PI controller.

1 Prerequisites

- C-887 controller with PI software CD (in the scope of delivery) and user documentation (short instructions, in the scope of delivery, or user manual, available from our website).
- PI hexapod with user documentation (short instructions, in the scope of delivery, or user manual, available from our website).
- You have installed the C-887 controller and the hexapod according to the installation instructions in the user documentation.
- You have executed test motions of the hexapod according to the instructions in the user documentation of the C-887 controller.
- TwinCAT 3.1 XAE
- You have copied the EtherCAT Device Description file (EDS) for your controller from the \EDS directory of the PI software CD into the TwinCAT IO configuration directory (e.g. “C:\TwinCAT\3.1\Config\Io\EtherCAT”).

2 Set Up a new TwinCAT Project and add the Controller

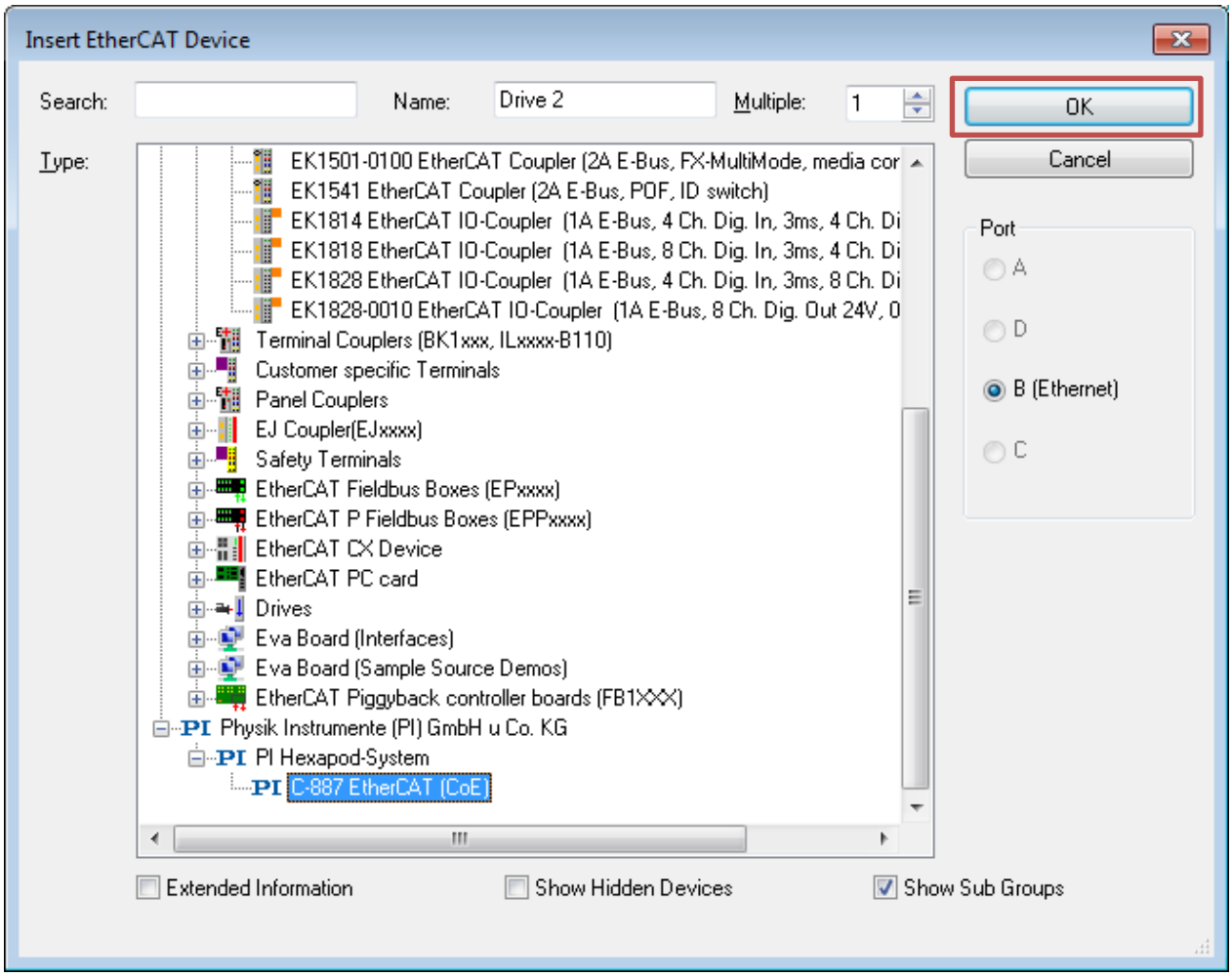
- Start with an empty TwinCAT project.
- Add your EtherCAT master device and adjust its adapter settings.



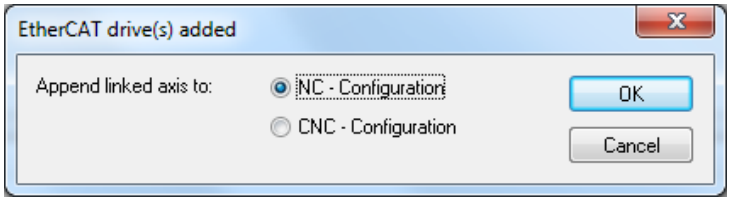
- To add a PI controller click the Scan button 

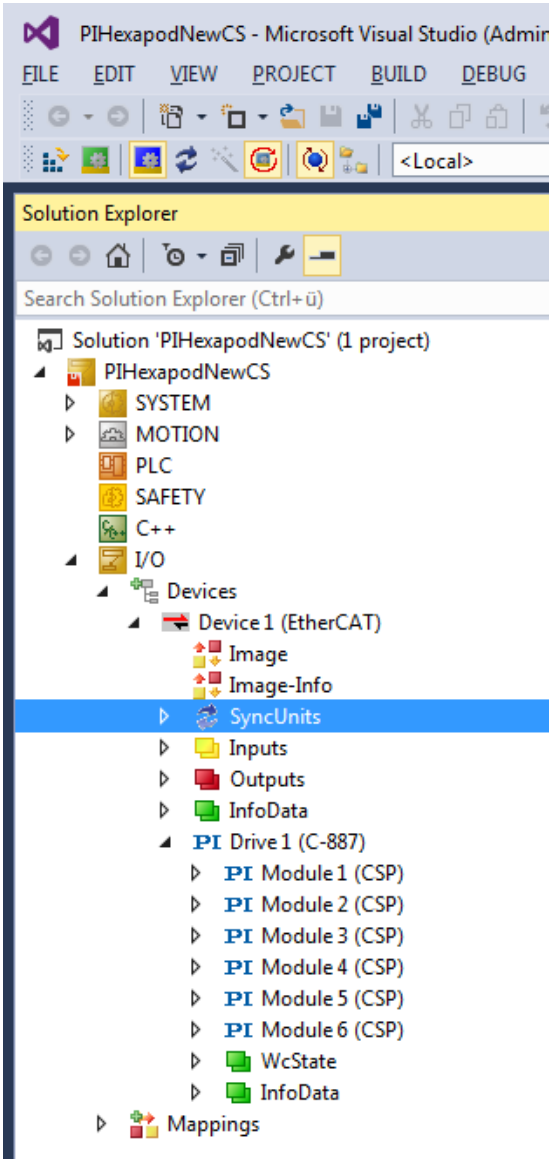
Or

- Add the controller manually as a new device (right-click at the EtherCAT master device → Add new item)



- Let TwinCAT automatically append the linked axes to a NC configuration





- Thus the position inputs and outputs of the PI controller modules are mapped automatically to the NC axes.
- For a manual mapping, map the position inputs and outputs of the axis to the corresponding NC axis as follows:

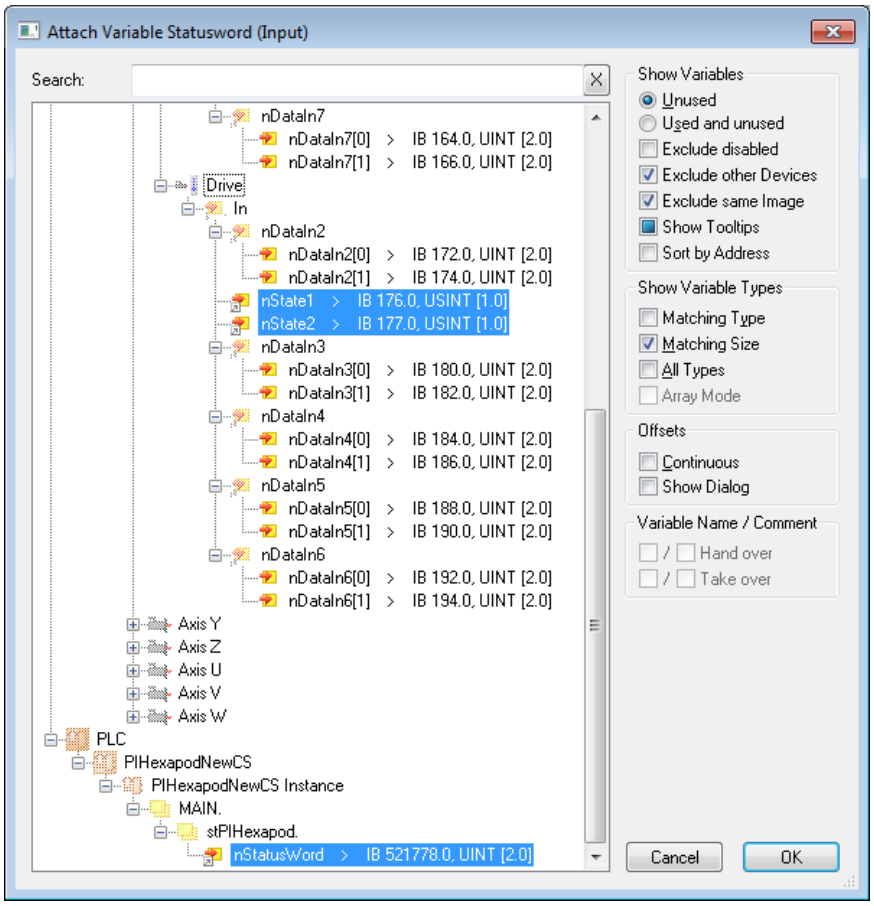
2.1 Position Inputs

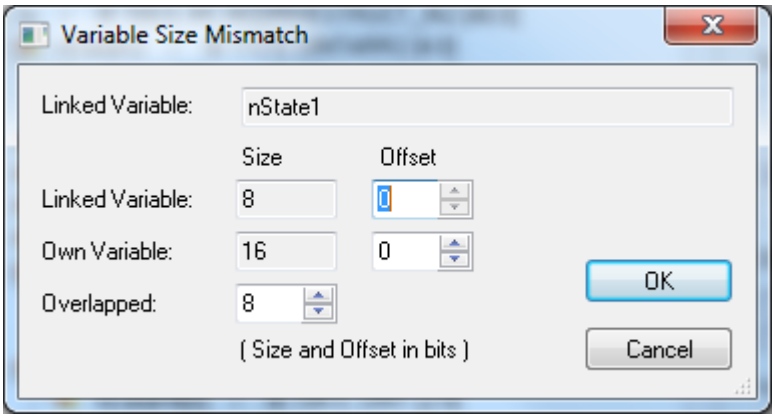
Statusword:

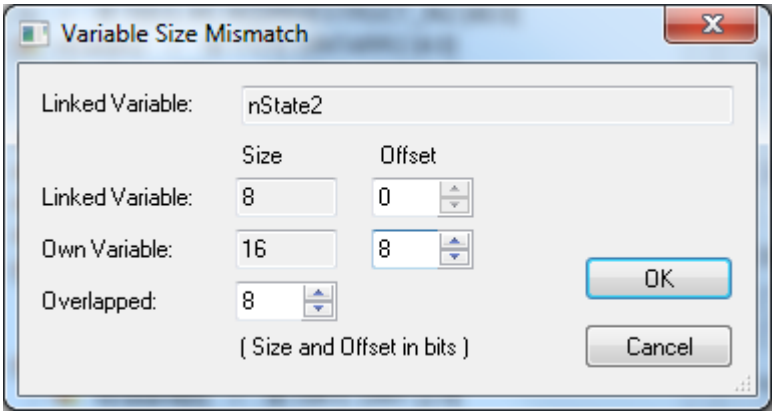
MOTION -> NC-Task 1 SAF -> Axes -> Axis X -> Drive -> In -> nState1 and nState2. Do this for each axis.
PLC -> PIHexapodNewCS -> PIHexapodNewCS Instance -> Main.stPIHexapod.nStatusWord

(press and hold Ctrl to select more than one item)

For nState2 set an offset of 8 bits.

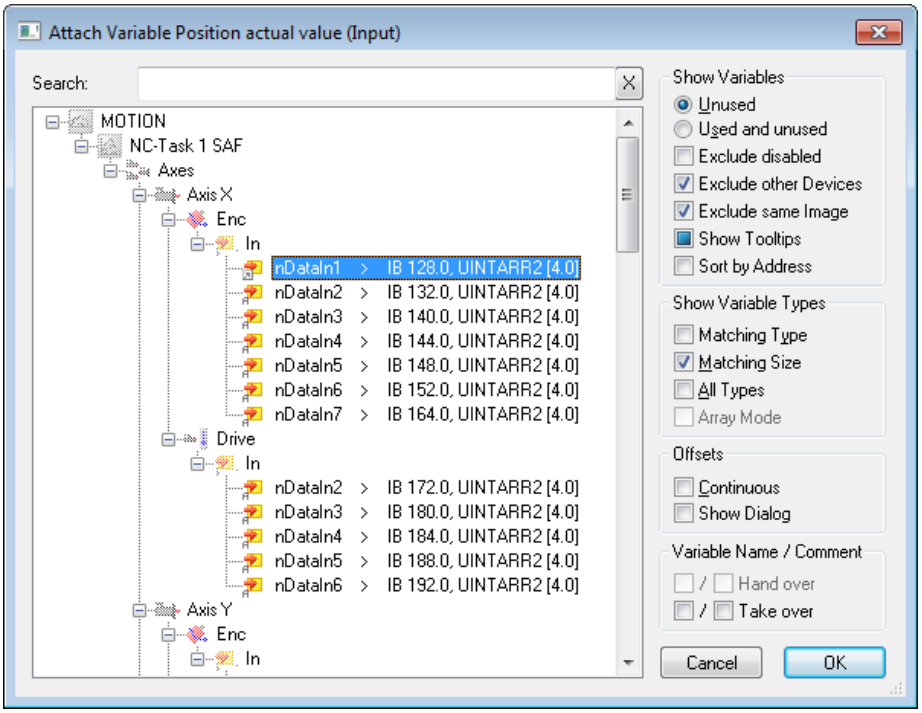

Attach Variable Statusword (Input)
Search:
nDataIn7
nDataIn7[0] > IB 164.0, UINT [2.0]
nDataIn7[1] > IB 166.0, UINT [2.0]
Drive
In
nDataIn2
nDataIn2[0] > IB 172.0, UINT [2.0]
nDataIn2[1] > IB 174.0, UINT [2.0]
nState1 > IB 176.0, USINT [1.0]
nState2 > IB 177.0, USINT [1.0]
nDataIn3
nDataIn3[0] > IB 180.0, UINT [2.0]
nDataIn3[1] > IB 182.0, UINT [2.0]
nDataIn4
nDataIn4[0] > IB 184.0, UINT [2.0]
nDataIn4[1] > IB 186.0, UINT [2.0]
nDataIn5
nDataIn5[0] > IB 188.0, UINT [2.0]
nDataIn5[1] > IB 190.0, UINT [2.0]
nDataIn6
nDataIn6[0] > IB 192.0, UINT [2.0]
nDataIn6[1] > IB 194.0, UINT [2.0]
Axis Y
Axis Z
Axis U
Axis V
Axis W
PLC
PIHexapodNewCS
PIHexapodNewCS Instance
MAIN.
stPIHexapod.
nStatusWord > IB 521778.0, UINT [2.0]
Show Variables
Unused
Used and unused
Exclude disabled
Exclude other Devices
Exclude same Image
Show Tooltips
Sort by Address
Show Variable Types
Matching Type
Matching Size
All Types
Array Mode
Offsets
Continuous
Show Dialog
Variable Name / Comment
/ Hand over
/ Take over
Cancel
OK


Variable Size Mismatch
Linked Variable:
nState1
Size
Offset
Linked Variable:
8
0
Own Variable:
16
0
Overlapped:
8
(Size and Offset in bits)
OK
Cancel


Variable Size Mismatch
Linked Variable:
nState2
Size
Offset
Linked Variable:
8
0
Own Variable:
16
8
Overlapped:
8
(Size and Offset in bits)
OK
Cancel

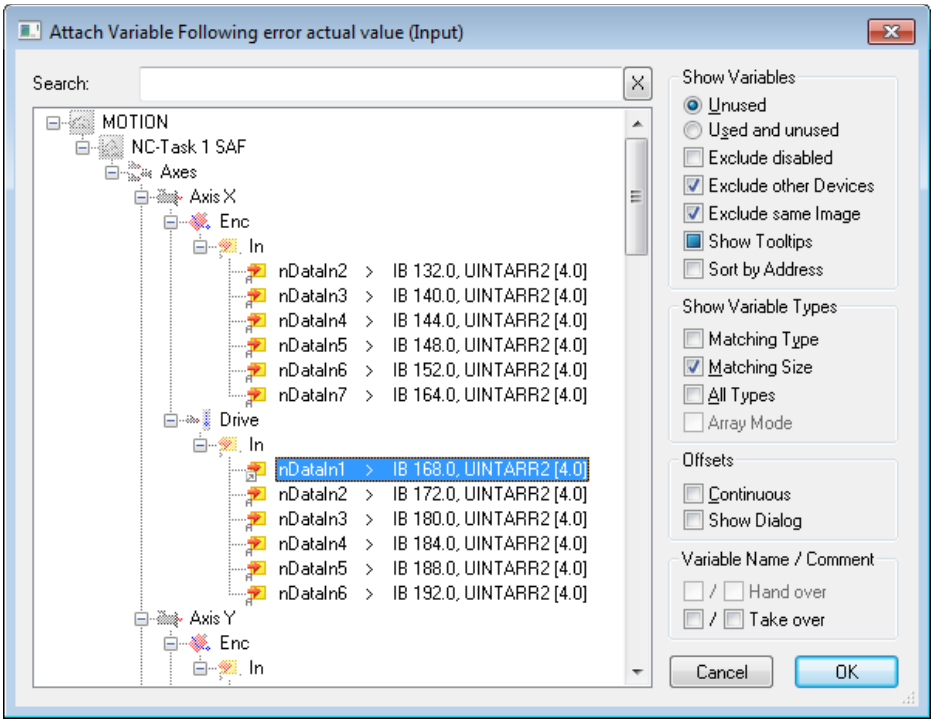
Position actual value:

MOTION -> NC-Task 1 SAF -> Axes -> Axis X -> Enc -> In -> nDataIn1. Do this for each axis.



Following error actual value:

MOTION -> NC-Task 1 SAF -> Axes -> Axis X -> Drive -> In -> nDataIn1. Do this for each axis.

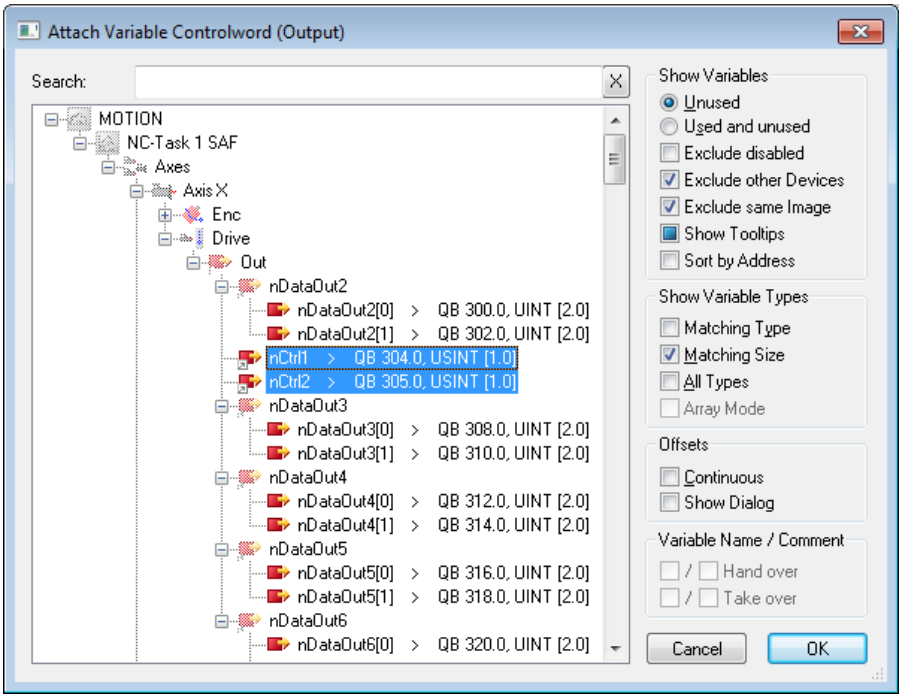


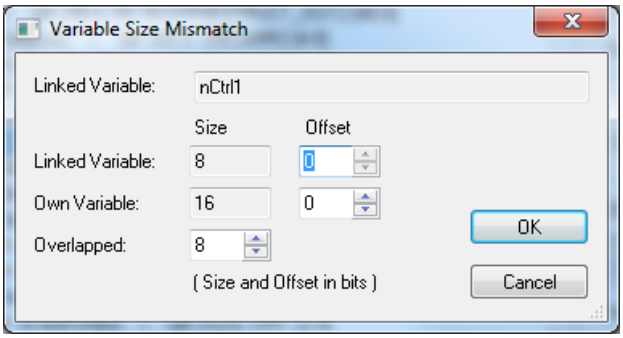
2.2 Position Outputs

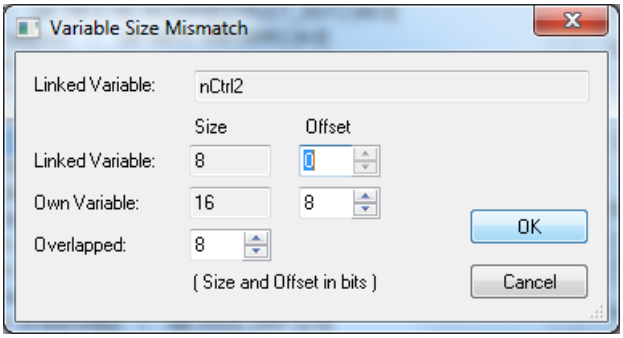
Controlword:

MOTION -> NC-Task 1 SAF -> Axes -> Axis X -> Drive -> Out -> nCtrl1 and nCtrl2. Do this for each axis.

For nCtrl2 set an offset of 8 bits.

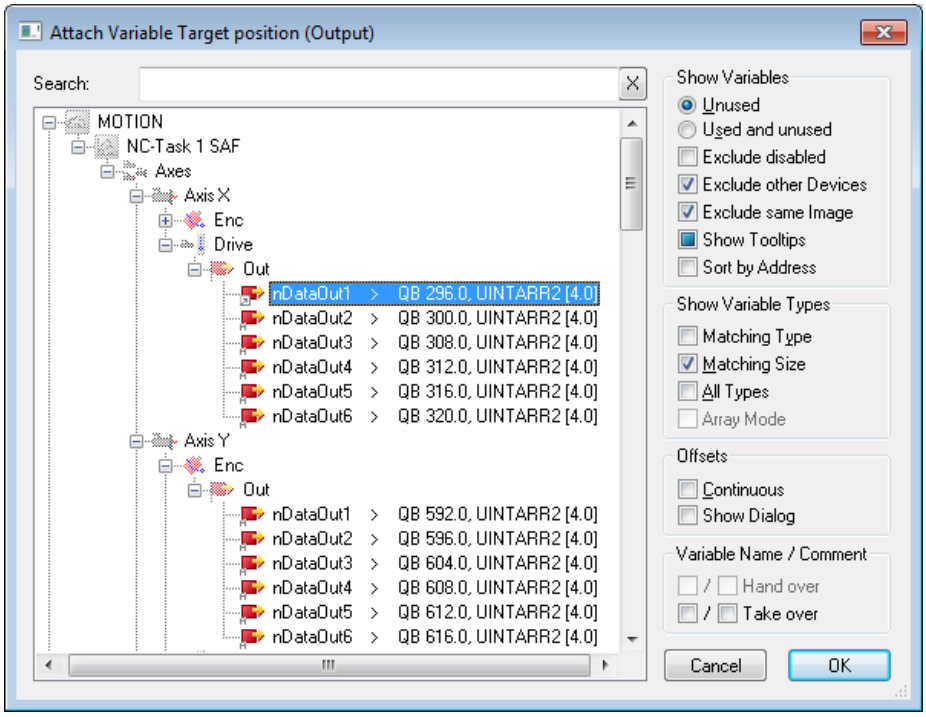






Target Position:

MOTION -> NC-Task 1 SAF -> Axes -> Axis X -> Drive -> Out -> nDataOut1. Do this for each axis.

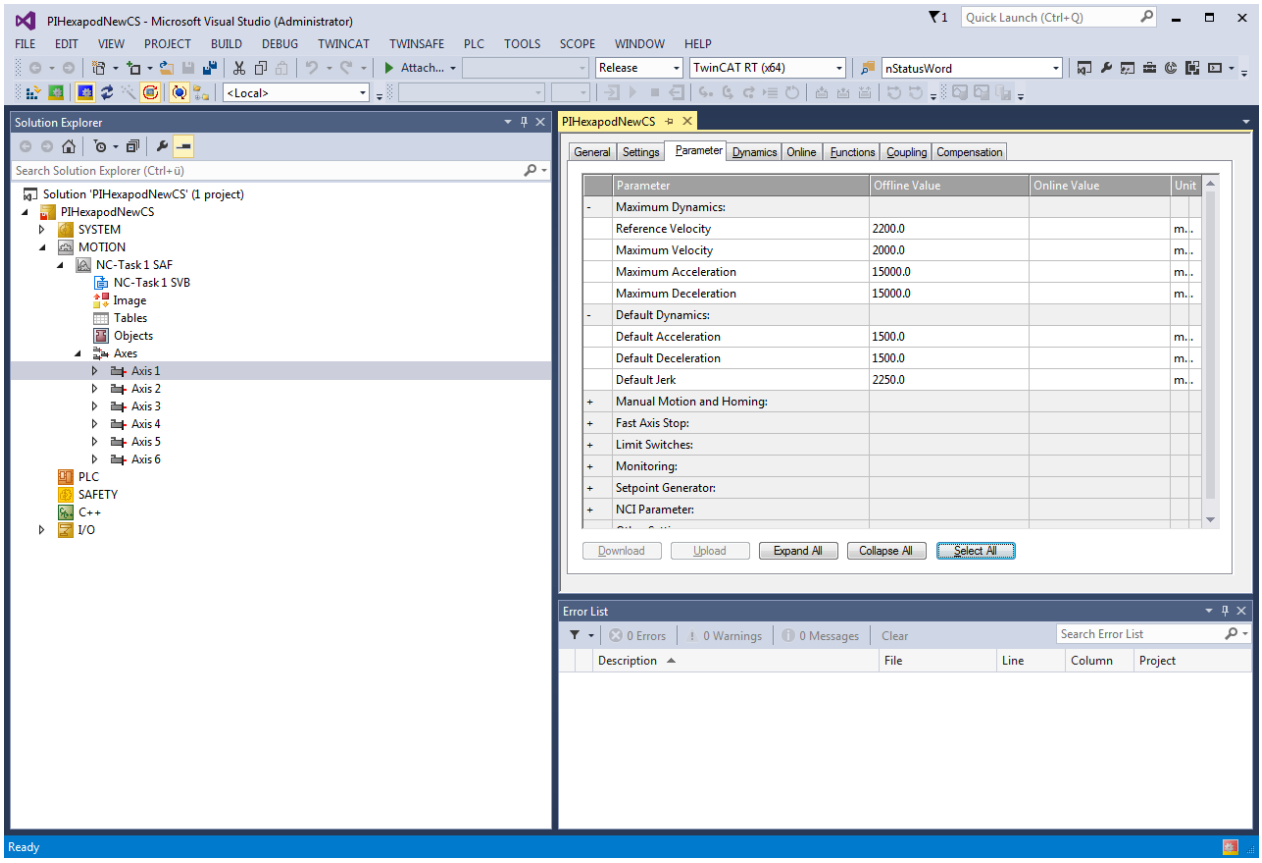


3 Adjust Axis Parameters

- Now adjust the NC axis parameter configuration as follows:

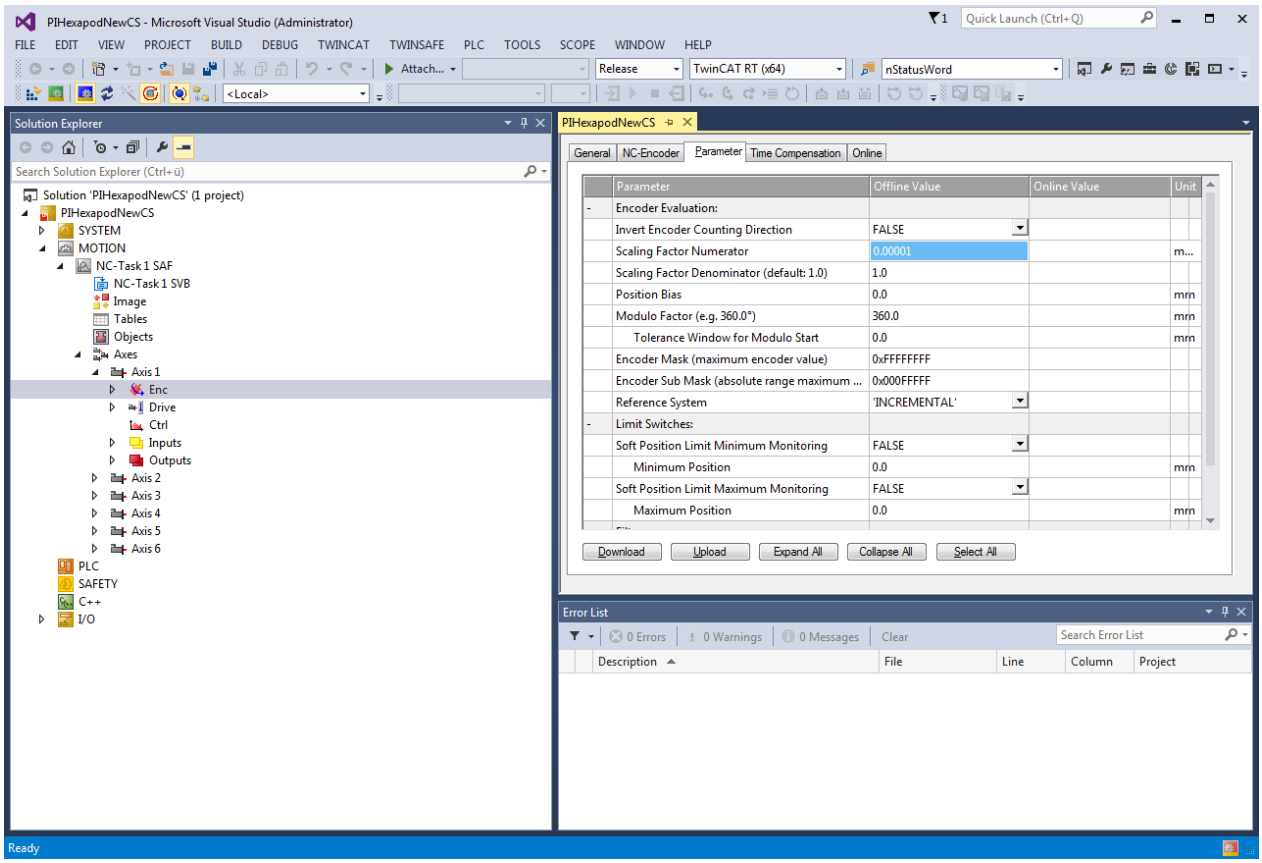
3.1 Axis Velocity and Acceleration

Adjust the velocity and acceleration according to the documentation of the hexapod. Do this for each axis.



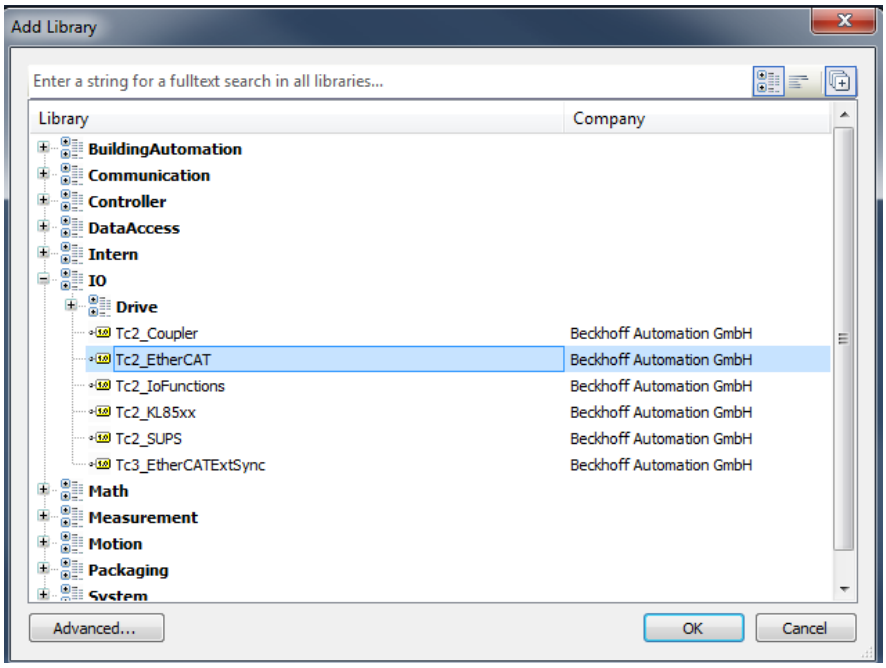
3.2 Encoder Parameter for Axes

Set *Scaling Factor* and *Reference System* of NC axes to suitable values (for details, see controller documentation). Do this for each axis.



4 Create a PLC Project for a Simple PTP Motion

- Create a new standard PLC project.
- Add the following libraries to the references section of the PLC project:
  IO -> Tc2_EtherCAT
  Motion -> PTP -> Tc2_MC2
  Motion -> NCI -> Tc2_NCI
  Motion -> NCI -> Tc2_PlcInterpolation
  System -> Tc2_Utilities



- Create a new Data Unit Type (DUT) with the identifier “ST_PIHexapod” as structure and add the following variables:

```
TYPE ST_PIHexapod :
STRUCT
   stX                              : AXIS_REF;
   stY                              : AXIS_REF;
   stZ                              : AXIS_REF;
   stU                              : AXIS_REF;
   stV                              : AXIS_REF;
   stW                              : AXIS_REF;
   nModeOfOperation         AT %Q*  : BYTE := 0;
   nModeOfOperationDisplay  AT %I*  : BYTE := 0;
   nStatusWord              AT %I*  : UINT := 0;
   WorkCSPDO                AT %Q*  : ARRAY [1..6] OF DINT := [0, 0, 0, 0, 0, 0];
   ToolCSPDO                AT %Q*  : ARRAY [1..6] OF DINT := [0, 0, 0, 0, 0, 0];
   bHomed                           : BOOL := FALSE;
   bEnabled                         : BOOL := FALSE;
END_STRUCT
END_TYPE
```

- Copy the following variables to the window for the local variables of the MAIN PLC program:

```
PROGRAM MAIN
VAR
   stPIHexapod                : ST_PIHexapod;
   fbPower                    : FB_PowerHexapod;
   fbHome                     : FB_HomeHexapod;
   fbSetAndActivateCSHexapod  : FB_SetAndActivateCSHexapod;
   fbMove                     : FB_MoveHexapod;
   nState                     : UDINT := 0;

   (* add "NetID" of EtherCAT Master and "EtherCAT address" of slave device *)
   NetID         : T_AmsNetId := '172.16.56.19.2.1';
```

```
   SlaveAddr       : UINT := 1001;
END_VAR
```

- Copy the following code to the MAIN PLC program (error handling has to be extended in the MAIN PLC program and each function block):

```
(*(c)2017-2019 Physik Instrumente (PI) GmbH & Co. KG
 Software products that are provided by PI are subject to the General Software License
Agreement of Physik Instrumente (PI) GmbH & Co. KG and may incorporate and/or make use of
third-party software components.
For more information, please read the General Software License Agreement and the Third
Party Software Note linked below.
General Software License Agreement:
http://www.physikinstrumente.com/download/EULA_PhysikInstrumenteGmbH_Co_KG.pdf
Third Party Software Note:
http://www.physikinstrumente.com/download/TPSWNote_PhysikInstrumenteGmbH_Co_KG.pdf

This sample was created with TwinCAT version v3.1.4022.4

PLC-axis parameters in mm, mm/s, mm/s2 and mm/s3
set scaling factor of NC-axes to suitable value (for details see controller documentation)

Sample program to demonstrate:
- homing of hexapod
- configuration of customized coordinate systems
- movement of hexapod axis

Error handling is missing and the sample is not complete to run a machine.

*)
stPIHexapod.stX.ReadStatus();
stPIHexapod.stY.ReadStatus();
stPIHexapod.stZ.ReadStatus();
stPIHexapod.stU.ReadStatus();
stPIHexapod.stV.ReadStatus();
stPIHexapod.stW.ReadStatus();

CASE nState OF

0:
   (* power on *)
   fbPower(
   bEnable:= TRUE,
   fOvr:= 100.0,
   stPIHexapod := stPIHexapod );

   IF stPIHexapod.bEnabled THEN
      nState  := 10;
   END_IF

10:
   (* homing *)
   fbHome.bExecute:= TRUE;

   IF fbHome.bDone THEN
      fbHome.bExecute:= FALSE;
      nState := 20;
   END_IF
```

```
20:
   (* reset coordinate system *)
   fbSetAndActivateCSHexapod.Work[1] := 0;
   fbSetAndActivateCSHexapod.Work[2] := 0;
   fbSetAndActivateCSHexapod.Work[3] := 0;
   fbSetAndActivateCSHexapod.Work[4] := 0;
   fbSetAndActivateCSHexapod.Work[5] := 0;
   fbSetAndActivateCSHexapod.Work[6] := 0;
   fbSetAndActivateCSHexapod.Tool[1] := 0;
   fbSetAndActivateCSHexapod.Tool[2] := 0;
   fbSetAndActivateCSHexapod.Tool[3] := 0;
   fbSetAndActivateCSHexapod.Tool[4] := 0;
   fbSetAndActivateCSHexapod.Tool[5] := 0;
   fbSetAndActivateCSHexapod.Tool[6] := 0;

   fbSetAndActivateCSHexapod.bExecute := TRUE;
   IF fbSetAndActivateCSHexapod.bDone THEN
      fbSetAndActivateCSHexapod.bExecute := FALSE;
      nState := 30;
   END_IF

30:
   (* move *)
   fbMove.PosX := 5;
   fbMove.bExecute := TRUE;

   IF fbMove.bDone THEN
      fbMove.bExecute := FALSE;
      nState:= 40;
   END_IF

40:
   (* move *)
   fbMove.PosX := 0;
   fbMove.bExecute := TRUE;

   IF fbMove.bDone THEN
      fbMove.bExecute := FALSE;
      nState:= 50;
   END_IF

50:
   (* activate coordinate system *)
   fbSetAndActivateCSHexapod.Work[1] := 0;
   fbSetAndActivateCSHexapod.Work[2] := 0;
   fbSetAndActivateCSHexapod.Work[3] := 0;
   fbSetAndActivateCSHexapod.Work[4] := 0;
   fbSetAndActivateCSHexapod.Work[5] := 0;
   fbSetAndActivateCSHexapod.Work[6] := 90;
   fbSetAndActivateCSHexapod.Tool[1] := 0;
   fbSetAndActivateCSHexapod.Tool[2] := 0;
   fbSetAndActivateCSHexapod.Tool[3] := 0;
   fbSetAndActivateCSHexapod.Tool[4] := 0;
   fbSetAndActivateCSHexapod.Tool[5] := 0;
   fbSetAndActivateCSHexapod.Tool[6] := 90;

   fbSetAndActivateCSHexapod.bExecute := TRUE;
```

```
   IF fbSetAndActivateCSHexapod.bDone THEN
      fbSetAndActivateCSHexapod.bExecute := FALSE;
      nState := 60;
   END_IF

60:
   (* move *)
   fbMove.PosX := 5;
   fbMove.bExecute := TRUE;

   IF fbMove.bDone THEN
      fbMove.bExecute := FALSE;
      nState:= 70;
   END_IF

70:
   (* move *)
   fbMove.PosX := 0;
   fbMove.bExecute := TRUE;

   IF fbMove.bDone THEN
      fbMove.bExecute := FALSE;
      nState:= 80;
   END_IF

80:
   nState := 20;

END_CASE

fbHome(stPIHexapod := stPIHexapod);
fbMove(stPIHexapod := stPIHexapod);
fbSetAndActivateCSHexapod(stPIHexapod   := stPIHexapod, sNetId := NetID, nSlaveAddr :=
SlaveAddr);
```

- Create a new PLC function block (Add -> POU…) named “FB_PowerHexapod” and copy the following variables to its window for the local variables:

```
FUNCTION_BLOCK FB_PowerHexapod
VAR_INPUT
   bEnable    : BOOL := FALSE;
   fOvr       : LREAL;
END_VAR
VAR_OUTPUT
   bError     : BOOL := FALSE;
   nErrorId   : UDINT := 0;
END_VAR
VAR_IN_OUT
   stPIHexapod   : ST_PIHexapod;
END_VAR
VAR
   fbPowerX      :  MC_Power;
   fbPowerY      :  MC_Power;
   fbPowerZ      :  MC_Power;
   fbPowerU      :  MC_Power;
   fbPowerV      :  MC_Power;
   fbPowerW      :  MC_Power;
END_VAR
```

- And copy the following code to the “FB_PowerHexapod” function block:

```
(* ToDo: extend error handling *)
(* Enable all six hexapod axes *)

fbPowerX(
   Axis:= stPIHexapod.stX ,
   Enable:= bEnable ,
   Enable_Positive:= bEnable,
   Enable_Negative:= bEnable,
   Override:=fOvr ,
   BufferMode:= ,
   Status=> ,
   Busy=> ,
   Active=> ,
   Error=> ,
   ErrorID=> );
fbPowerY(
   Axis:= stPIHexapod.stY ,
   Enable:= bEnable ,
   Enable_Positive:= bEnable,
   Enable_Negative:= bEnable,
   Override:=fOvr ,
   BufferMode:= ,
   Status=> ,
   Busy=> ,
   Active=> ,
   Error=> ,
   ErrorID=> );
fbPowerZ(
   Axis:= stPIHexapod.stZ ,
   Enable:= bEnable ,
   Enable_Positive:= bEnable,
   Enable_Negative:= bEnable,
   Override:=fOvr ,
   BufferMode:= ,
   Status=> ,
   Busy=> ,
   Active=> ,
   Error=> ,
   ErrorID=> );
fbPowerU(
   Axis:= stPIHexapod.stU ,
   Enable:= bEnable ,
   Enable_Positive:= bEnable,
   Enable_Negative:= bEnable,
   Override:=fOvr ,
   BufferMode:= ,
   Status=> ,
   Busy=> ,
   Active=> ,
   Error=> ,
   ErrorID=> );
fbPowerV(
   Axis:= stPIHexapod.stV ,
   Enable:= bEnable ,
   Enable_Positive:= bEnable,
   Enable_Negative:= bEnable,
```

```
   Override:=fOvr ,
   BufferMode:= ,
   Status=> ,
   Busy=> ,
   Active=> ,
   Error=> ,
   ErrorID=> );
fbPowerW(
   Axis:= stPIHexapod.stW ,
   Enable:= bEnable ,
   Enable_Positive:= bEnable,
   Enable_Negative:= bEnable,
   Override:=fOvr ,
   BufferMode:= ,
   Status=> ,
   Busy=> ,
   Active=> ,
   Error=> ,
   ErrorID=> );

IF fbPowerX.Error THEN
   bError := TRUE;
   nErrorId := fbPowerX.ErrorID;
ELSE
   IF fbPowerX.Status AND fbPowerY.Status AND fbPowerZ.Status AND fbPowerU.Status AND
fbPowerV.Status AND fbPowerW.Status THEN
      stPIHexapod.bEnabled := TRUE;
   ELSE
      stPIHexapod.bEnabled := FALSE;
   END_IF
END_IF
```

- Create a new PLC function block (Add -> POU…) named “FB_HomeHexapod” and copy the following variables to its window for the local variables:

```
FUNCTION_BLOCK FB_HomeHexapod
VAR_INPUT
   bExecute      : BOOL  := FALSE;
END_VAR
VAR_OUTPUT
   bBusy         : BOOL  := FALSE;
   bDone         : BOOL  := FALSE;
   bError        : BOOL  := FALSE;
   nErrorId      : UDINT := 0;
END_VAR
VAR_IN_OUT
   stPIHexapod   : ST_PIHexapod;
END_VAR
VAR
   nState        : UDINT := 0;
   fbSetPos      : FB_SetPosHexapod;
END_VAR
```

- And copy the following code to the “FB_HomeHexapod” function block:

```
   (*   ToDo: extend error handling  *)

CASE nState OF
```

```
0:
   IF bExecute THEN
      bBusy := TRUE;
      bDone := FALSE;
      nState  := 10;
   END_IF
10:
   (* set Mode of Operation to 6 and wait until Mode of Operation Display switches to 6 *)
   stPIHexapod.nModeOfOperation := 6;
   IF stPIHexapod.nModeOfOperationDisplay = 6 THEN
      nState := 20;
   END_IF

20:

   (* wait until homing move is finished *)
   (* Statusword has to be set as 1010000100111 *)
   IF (stPIHexapod.nStatusWord AND 5159) = 5159 THEN
      stPIHexapod.bHomed := TRUE;
      nState := 30;

   END_IF

30:
   (* set hexapod absolute positions to 0 *)
   fbSetPos.PosX := 0.0;
   fbSetPos.PosY := 0.0;
   fbSetPos.PosZ := 0.0;
   fbSetPos.PosU := 0.0;
   fbSetPos.PosV := 0.0;
   fbSetPos.PosW := 0.0;
   fbSetPos.Execute := TRUE;

   IF fbSetPos.bError THEN
      nState := 9999;
   ELSE
      IF (fbSetPos.bDone) THEN
         fbSetPos.Execute := FALSE;
         nState := 40;
      END_IF
   END_IF

40:
   (* set Mode of Operation to 8 and wait until Mode of Operation Display switches to 8 *)
   stPIHexapod.nModeOfOperation := 8;
   IF stPIHexapod.nModeOfOperationDisplay = 8 THEN
      nState := 50;
   END_IF

50:
   bBusy := FALSE;
   IF NOT bExecute THEN
      nState  := 0;
      bDone := FALSE;
   ELSE
      bDone := TRUE;
   END_IF
```

```
9999:
   bError := TRUE;
   nErrorId := fbSetPos.nErrorId;

END_CASE

fbSetPos (stPIHexapod := stPIHexapod);
```

- Create a new PLC function block (Add -> POU…) named “FB_MoveHexapod” and copy the following variables to its window for the local variables:

```
FUNCTION_BLOCK FB_MoveHexapod
VAR_IN_OUT
       stPIHexapod          : ST_PIHexapod;
END_VAR
VAR_INPUT
       bExecute             : BOOL := FALSE;
       PosX                 : LREAL := 0;
       PosY                 : LREAL := 0;
       PosZ                 : LREAL := 0;
       PosU                 : LREAL := 0;
       PosV                 : LREAL := 0;
       PosW                 : LREAL := 0 ;

       (* default parameters, please adapt according to hexapod type! *)
       VelX                 : LREAL := 1;
       VelY                 : LREAL := 1;
       VelZ                 : LREAL := 1;
       VelU                 : LREAL := 1;
       VelV                 : LREAL := 1;
       VelW                 : LREAL := 1;
       AccX                 : LREAL := 10;
       AccY                 : LREAL := 10;
       AccZ                 : LREAL := 10;
       AccU                 : LREAL := 10;
       AccV                 : LREAL := 10;
       AccW                 : LREAL := 10;
END_VAR
VAR_OUTPUT
       bBusy                : BOOL := FALSE;
       bDone                : BOOL := FALSE;
       bError               : BOOL := FALSE;
       nErrorId             : UDINT      := 0;
END_VAR
VAR
       fbMovX               :      MC_MoveAbsolute;
       fbMovY               :      MC_MoveAbsolute;
       fbMovZ               :      MC_MoveAbsolute;
       fbMovU               :      MC_MoveAbsolute;
       fbMovV               :      MC_MoveAbsolute;
       fbMovW               :      MC_MoveAbsolute;

       nState               : UDINT      := 0;
END_VAR
```

- And copy the following code to the “FB_MoveHexapod” function block:

```
(* ToDo: extend error handling *)
```

```
CASE nState OF
0:
	IF bExecute THEN
		bBusy  := TRUE;
		bDone  := FALSE;
		nState := 10;
	END_IF

10:
	IF VelX < 0.1 THEN
		VelX := 1;
	END_IF
	IF AccX < 0.1 THEN
		AccX := 1;
	END_IF
	IF VelY < 0.1 THEN
		VelY := 1;
	END_IF
	IF AccY < 0.1 THEN
		AccY := 1;
	END_IF
	IF VelZ < 0.1 THEN
		VelZ := 1;
	END_IF
	IF AccZ < 0.1 THEN
		AccZ := 1;
	END_IF
	IF VelU < 0.1 THEN
		VelU := 1;
	END_IF
	IF AccU < 0.1 THEN
		AccU := 1;
	END_IF
	IF VelV < 0.1 THEN
		VelV := 1;
	END_IF
	IF AccV < 0.1 THEN
		AccV := 1;
	END_IF
	IF VelW < 0.1 THEN
		VelW := 1;
	END_IF
	IF AccW < 0.1 THEN
		AccW := 1;
	END_IF

	fbMovX.Position:= PosX;
	fbMovX.Velocity:= VelX;
	fbMovX.Acceleration:= AccX;
	fbMovX.Execute:= bExecute;

	fbMovY.Position:= PosY;
	fbMovY.Velocity:= VelY;
	fbMovY.Acceleration:= AccY;
	fbMovY.Execute:= bExecute;

	fbMovZ.Position:= PosZ;
```

```
	fbMovZ.Velocity:= VelZ;
	fbMovZ.Acceleration:= AccZ;
	fbMovZ.Execute:= bExecute;

	fbMovU.Position:= PosU;
	fbMovU.Velocity:= VelU;
	fbMovU.Acceleration:= AccU;
	fbMovU.Execute:= bExecute;

	fbMovV.Position:= PosV;
	fbMovV.Velocity:= VelV;
	fbMovV.Acceleration:= AccV;
	fbMovV.Execute:= bExecute;

	fbMovW.Position:= PosW;
	fbMovW.Velocity:= VelW;
	fbMovW.Acceleration:= AccW;
	fbMovW.Execute:= bExecute;

	IF fbMovX.Done AND fbMovY.Done AND fbMovZ.Done AND fbMovU.Done AND fbMovV.Done AND
fbMovW.Done THEN
		fbMovX.Execute:= FALSE;
		fbMovY.Execute:= FALSE;
		fbMovZ.Execute:= FALSE;
		fbMovU.Execute:= FALSE;
		fbMovV.Execute:= FALSE;
		fbMovW.Execute:= FALSE;

		nState := 500;
	END_IF

	IF fbMovX.Error OR fbMovY.Error OR fbMovZ.Error OR fbMovU.Error OR fbMovV.Error OR
fbMovW.Error THEN
		fbMovX.Execute:= FALSE;
		fbMovY.Execute:= FALSE;
		fbMovZ.Execute:= FALSE;
		fbMovU.Execute:= FALSE;
		fbMovV.Execute:= FALSE;
		fbMovW.Execute:= FALSE;

		nState := 9999;
	END_IF

500:
	bBusy := FALSE;
	IF NOT bExecute THEN
		nState := 0;
		bDone := FALSE;
	ELSE
		bDone := TRUE;
	END_IF

9999:
	(* an error occurred *)
	(* error handling is still missing *)
	bError		:= TRUE;
	nErrorId	:= fbMovX.ErrorId;
```

```
END_CASE

fbMovX( Axis:=stPIHexapod.stX );
fbMovY( Axis:=stPIHexapod.stY );
fbMovZ( Axis:=stPIHexapod.stZ );
fbMovU( Axis:=stPIHexapod.stU );
fbMovV( Axis:=stPIHexapod.stV );
fbMovW( Axis:=stPIHexapod.stW );
```

- Create a new PLC function block (Add -> POU…) named “FB_SetAndActivateCSHexapod” and copy the following variables to its window for the local variables:

```
FUNCTION_BLOCK FB_SetAndActivateCSHexapod
VAR_IN_OUT
       stPIHexapod          : ST_PIHexapod;
END_VAR
VAR_INPUT
       bExecute             : BOOL := FALSE;
       sNetId          : T_AmsNetId;               (* NetId of EtherCAT Master *)
    nSlaveAddr      : UINT;                     (* Port Number of EtherCAT Slave *)
       Work                 : ARRAY [1..6] OF LREAL;
       Tool                 : ARRAY [1..6] OF LREAL;
END_VAR
VAR_OUTPUT
       bBusy                : BOOL := FALSE;
       bDone                : BOOL := FALSE;
       bError               : BOOL := FALSE;
       nErrorId             : UDINT       := 0;
END_VAR
VAR
       nState               : UDINT       := 0;
       nHsk                 : DINT := 1;
       bufferWork           : ARRAY [1..6] OF DINT;
       bufferTool           : ARRAY [1..6] OF DINT;
       i                          : INT;

       fbSdoWriteWorkX      : FB_EcCoESdoWrite;
       fbSdoWriteWorkY      : FB_EcCoESdoWrite;
       fbSdoWriteWorkZ      : FB_EcCoESdoWrite;
       fbSdoWriteWorkU      : FB_EcCoESdoWrite;
       fbSdoWriteWorkV      : FB_EcCoESdoWrite;
       fbSdoWriteWorkW      : FB_EcCoESdoWrite;
       fbSdoWriteToolX      : FB_EcCoESdoWrite;
       fbSdoWriteToolY      : FB_EcCoESdoWrite;
       fbSdoWriteToolZ      : FB_EcCoESdoWrite;
       fbSdoWriteToolU      : FB_EcCoESdoWrite;
       fbSdoWriteToolV      : FB_EcCoESdoWrite;
       fbSdoWriteToolW      : FB_EcCoESdoWrite;
       fbSdoWriteHsk    : FB_EcCoESdoWrite;

       fbSetPos             : FB_SetPosHexapod;

       fbTimer1: TON;
END_VAR
```

- And copy the following code to the “FB_SetAndActivateCSHexapod” function block:

```
(* ToDo: extend error handling *)

CASE nState OF
0:
	IF bExecute THEN
		bBusy  := TRUE;
		bDone  := FALSE;
		nState := 10;
		fbTimer1(PT:=T#100MS);
	END_IF

10:
	(* set Mode of Operation to 0 and wait until Mode of Operation Display switches to 0
*)
	stPIHexapod.nModeOfOperation := 0;
	IF stPIHexapod.nModeOfOperationDisplay = 0 THEN
		nState := 20;
	END_IF

20:
	FOR i := 1 TO 6 DO
		bufferWork[i] := LREAL_TO_DINT(Work[i] * 1000);
		bufferTool[i] := LREAL_TO_DINT(Tool[i] * 1000);
	END_FOR

	nState := 30;

30:
	(* write Work coordinate system to SDOs *)
	fbSdoWriteWorkX(
			sNetId      := sNetId,
			nSlaveAddr   := nSlaveAddr,
			nIndex      := 16#5000,
			nSubIndex   := 1,
			pSrcBuf     := ADR(bufferWork[1]),
			cbBufLen    := SIZEOF(bufferWork[1]),
			bExecute    := bExecute
	);
	fbSdoWriteWorkY(
			sNetId      := sNetId,
			nSlaveAddr   := nSlaveAddr,
			nIndex      := 16#5000,
			nSubIndex   := 2,
			pSrcBuf     := ADR(bufferWork[2]),
			cbBufLen    := SIZEOF(bufferWork[2]),
			bExecute    := bExecute
	);
	fbSdoWriteWorkZ(
			sNetId      := sNetId,
			nSlaveAddr   := nSlaveAddr,
			nIndex      := 16#5000,
			nSubIndex   := 3,
			pSrcBuf     := ADR(bufferWork[3]),
			cbBufLen    := SIZEOF(bufferWork[3]),
			bExecute    := bExecute
	);
```

```
	fbSdoWriteWorkU(
			sNetId		:= sNetId,
			nSlaveAddr	:= nSlaveAddr,
			nIndex		:= 16#5000,
			nSubIndex	:= 4,
			pSrcBuf		:= ADR(bufferWork[4]),
			cbBufLen	:= SIZEOF(bufferWork[4]),
			bExecute	:= bExecute
	);
	fbSdoWriteWorkV(
			sNetId		:= sNetId,
			nSlaveAddr	:= nSlaveAddr,
			nIndex		:= 16#5000,
			nSubIndex	:= 5,
			pSrcBuf		:= ADR(bufferWork[5]),
			cbBufLen	:= SIZEOF(bufferWork[5]),
			bExecute	:= bExecute
	);
	fbSdoWriteWorkW(
			sNetId		:= sNetId,
			nSlaveAddr	:= nSlaveAddr,
			nIndex		:= 16#5000,
			nSubIndex	:= 6,
			pSrcBuf		:= ADR(bufferWork[6]),
			cbBufLen	:= SIZEOF(bufferWork[6]),
			bExecute	:= bExecute
	);

	IF NOT fbSdoWriteWorkX.bBusy AND NOT fbSdoWriteWorkY.bBusy AND NOT
fbSdoWriteWorkZ.bBusy AND NOT fbSdoWriteWorkU.bBusy AND NOT fbSdoWriteWorkV.bBusy AND NOT
fbSdoWriteWorkW.bBusy THEN
		IF NOT (fbSdoWriteWorkX.bError OR fbSdoWriteWorkY.bError OR
fbSdoWriteWorkZ.bError OR fbSdoWriteWorkU.bError OR fbSdoWriteWorkV.bError OR
fbSdoWriteWorkW.bError) THEN
				(* write successful *)
				bError := FALSE;
				nErrorId := 0;
				nState:= 40;
		ELSE
				(* write failed *)
				bError := TRUE;
				IF fbSdoWriteWorkX.bError THEN
					nErrorId := fbSdoWriteWorkX.nErrId;
				ELSIF fbSdoWriteWorkY.bError THEN
					nErrorId := fbSdoWriteWorkY.nErrId;
				ELSIF fbSdoWriteWorkZ.bError THEN
					nErrorId := fbSdoWriteWorkZ.nErrId;
				ELSIF fbSdoWriteWorkU.bError THEN
					nErrorId := fbSdoWriteWorkU.nErrId;
				ELSIF fbSdoWriteWorkV.bError THEN
					nErrorId := fbSdoWriteWorkV.nErrId;
				ELSIF fbSdoWriteWorkW.bError THEN
					nErrorId := fbSdoWriteWorkW.nErrId;
				END_IF
				nState := 9999;
		END_IF

		fbSdoWriteWorkX(bExecute := FALSE);
```

```
            fbSdoWriteWorkY(bExecute := FALSE);
            fbSdoWriteWorkZ(bExecute := FALSE);
            fbSdoWriteWorkU(bExecute := FALSE);
            fbSdoWriteWorkV(bExecute := FALSE);
            fbSdoWriteWorkW(bExecute := FALSE);
      END_IF

40:
      (* write Tool coordinate system to SDOs *)
      fbSdoWriteToolX(
                  sNetId      := sNetId,
                  nSlaveAddr   := nSlaveAddr,
                  nIndex      := 16#5001,
                  nSubIndex   := 1,
                  pSrcBuf     := ADR(bufferTool[1]),
                  cbBufLen    := SIZEOF(bufferTool[1]),
                  bExecute    := bExecute
      );
      fbSdoWriteToolY(
                  sNetId      := sNetId,
                  nSlaveAddr   := nSlaveAddr,
                  nIndex      := 16#5001,
                  nSubIndex   := 2,
                  pSrcBuf     := ADR(bufferTool[2]),
                  cbBufLen    := SIZEOF(bufferTool[2]),
                  bExecute    := bExecute
      );
      fbSdoWriteToolZ(
                  sNetId      := sNetId,
                  nSlaveAddr   := nSlaveAddr,
                  nIndex      := 16#5001,
                  nSubIndex   := 3,
                  pSrcBuf     := ADR(bufferTool[3]),
                  cbBufLen    := SIZEOF(bufferTool[3]),
                  bExecute    := bExecute
      );
      fbSdoWriteToolU(
                  sNetId      := sNetId,
                  nSlaveAddr   := nSlaveAddr,
                  nIndex      := 16#5001,
                  nSubIndex   := 4,
                  pSrcBuf     := ADR(bufferTool[4]),
                  cbBufLen    := SIZEOF(bufferTool[4]),
                  bExecute    := bExecute
      );
      fbSdoWriteToolV(
                  sNetId      := sNetId,
                  nSlaveAddr   := nSlaveAddr,
                  nIndex      := 16#5001,
                  nSubIndex   := 5,
                  pSrcBuf     := ADR(bufferTool[5]),
                  cbBufLen    := SIZEOF(bufferTool[5]),
                  bExecute    := bExecute
      );
      fbSdoWriteToolW(
                  sNetId      := sNetId,
                  nSlaveAddr   := nSlaveAddr,
                  nIndex      := 16#5001,
```

```
                    nSubIndex   := 6,
                    pSrcBuf     := ADR(bufferTool[6]),
                    cbBufLen    := SIZEOF(bufferTool[6]),
                    bExecute    := bExecute
      );

      IF NOT fbSdoWriteToolX.bBusy AND NOT fbSdoWriteToolY.bBusy AND NOT
fbSdoWriteToolZ.bBusy AND NOT fbSdoWriteToolU.bBusy AND NOT fbSdoWriteToolV.bBusy AND NOT
fbSdoWriteToolW.bBusy THEN
            IF NOT (fbSdoWriteToolX.bError OR fbSdoWriteToolY.bError OR
fbSdoWriteToolZ.bError OR fbSdoWriteToolU.bError OR fbSdoWriteToolV.bError OR
fbSdoWriteToolW.bError) THEN
                        (* write successful *)
                        bError := FALSE;
                        nErrorId := 0;
                        nState:= 50;
            ELSE
                        (* write failed *)
                        bError := TRUE;
                        IF fbSdoWriteToolX.bError THEN
                              nErrorId := fbSdoWriteToolX.nErrId;
                        ELSIF fbSdoWriteToolY.bError THEN
                              nErrorId := fbSdoWriteToolY.nErrId;
                        ELSIF fbSdoWriteToolZ.bError THEN
                              nErrorId := fbSdoWriteToolZ.nErrId;
                        ELSIF fbSdoWriteToolU.bError THEN
                              nErrorId := fbSdoWriteToolU.nErrId;
                        ELSIF fbSdoWriteToolV.bError THEN
                              nErrorId := fbSdoWriteToolV.nErrId;
                        ELSIF fbSdoWriteToolW.bError THEN
                              nErrorId := fbSdoWriteToolW.nErrId;
                        END_IF
                        nState := 9999;
            END_IF

            fbSdoWriteToolX(bExecute := FALSE);
            fbSdoWriteToolY(bExecute := FALSE);
            fbSdoWriteToolZ(bExecute := FALSE);
            fbSdoWriteToolU(bExecute := FALSE);
            fbSdoWriteToolV(bExecute := FALSE);
            fbSdoWriteToolW(bExecute := FALSE);
      END_IF


50:
      (* write handshake for Work CS to SDOs *)
      nHsk := 1;

      fbSdoWriteHsk(
                    sNetId      := sNetId,
                    nSlaveAddr   := nSlaveAddr,
                    nIndex      := 16#5010,
                    nSubIndex   := 1,
                    pSrcBuf     := ADR(nHsk),
                    cbBufLen    := SIZEOF(nHsk),
                    bExecute    := bExecute
      );
```

```
	IF NOT fbSdoWriteHsk.bBusy THEN
		IF NOT fbSdoWriteHsk.bError THEN
				(* write successful *)
				bError := FALSE;
				nErrorId := 0;
				nState:= 60;
		ELSE
				(* write failed *)
				bError := TRUE;
				IF fbSdoWriteHsk.bError THEN
					nErrorId := fbSdoWriteHsk.nErrId;
				END_IF
				nState := 9999;
		END_IF

		fbSdoWriteHsk(bExecute := FALSE);
	END_IF

60:
	(* write handshake for Tool CS to SDOs *)
	nHsk := 1;

	fbSdoWriteHsk(
			sNetId		:= sNetId,
			nSlaveAddr	:= nSlaveAddr,
			nIndex		:= 16#5010,
			nSubIndex	:= 2,
			pSrcBuf		:= ADR(nHsk),
			cbBufLen	:= SIZEOF(nHsk),
			bExecute	:= bExecute
	);

	IF NOT fbSdoWriteHsk.bBusy THEN
		IF NOT fbSdoWriteHsk.bError THEN
				(* write successful *)
				bError := FALSE;
				nErrorId := 0;
				nState:= 70;
		ELSE
				(* write failed *)
				nState := 9999;
		END_IF

		fbSdoWriteHsk(bExecute := FALSE);
	END_IF

70:
	(* wait for click, slave controller needs some time to finish internal calculations
*)
	fbTimer1.IN := TRUE;
	IF fbTimer1.Q THEN
		nState := 80;
		fbTimer1.IN := FALSE;
	END_IF

80:
	(* set position to new actual position *)
	fbSetPos.PosX := stPIHexapod.stX.NcToPlc.ActPos;
```

```
        fbSetPos.PosY := stPIHexapod.stY.NcToPlc.ActPos;
        fbSetPos.PosZ := stPIHexapod.stZ.NcToPlc.ActPos;
        fbSetPos.PosU := stPIHexapod.stU.NcToPlc.ActPos;
        fbSetPos.PosV := stPIHexapod.stV.NcToPlc.ActPos;
        fbSetPos.PosW := stPIHexapod.stW.NcToPlc.ActPos;
        fbSetPos.bClearPositionLag := TRUE;
        fbSetPos.Execute := TRUE;

        IF (fbSetPos.bDone) THEN
                fbSetPos.Execute := FALSE;
                nState := 90;
        END_IF

90:
        (* set Mode of Operation to 8 and wait until Mode of Operation Display switches to 8
*)
        stPIHexapod.nModeOfOperation := 8;
        IF stPIHexapod.nModeOfOperationDisplay = 8 THEN
                nState := 100;
        END_IF

100:
        bBusy := FALSE;
        IF NOT bExecute THEN
                nState := 0;
                bDone := FALSE;
        ELSE
                bDone := TRUE;
        END_IF

9999:
        (* an error occured *)
        bError := TRUE;
        IF fbSdoWriteHsk.bError THEN
                nErrorId := fbSdoWriteHsk.nErrId;
        END_IF

END_CASE

fbSetPos (stPIHexapod := stPIHexapod);
fbTimer1();
```

- Create a new PLC function block (Add -> POU…) named “FB_SetPosHexapod” and copy the following variables to its window for the local variables:

```
FUNCTION_BLOCK FB_SetPosHexapod
VAR_IN_OUT
        stPIHexapod          : ST_PIHexapod;
END_VAR
VAR_INPUT
        Execute                     : BOOL := FALSE;
        PosX                 : LREAL := 0.0;
        PosY                 : LREAL := 0.0;
        PosZ                 : LREAL := 0.0;
        PosU                 : LREAL := 0.0;
        PosV                 : LREAL := 0.0;
```

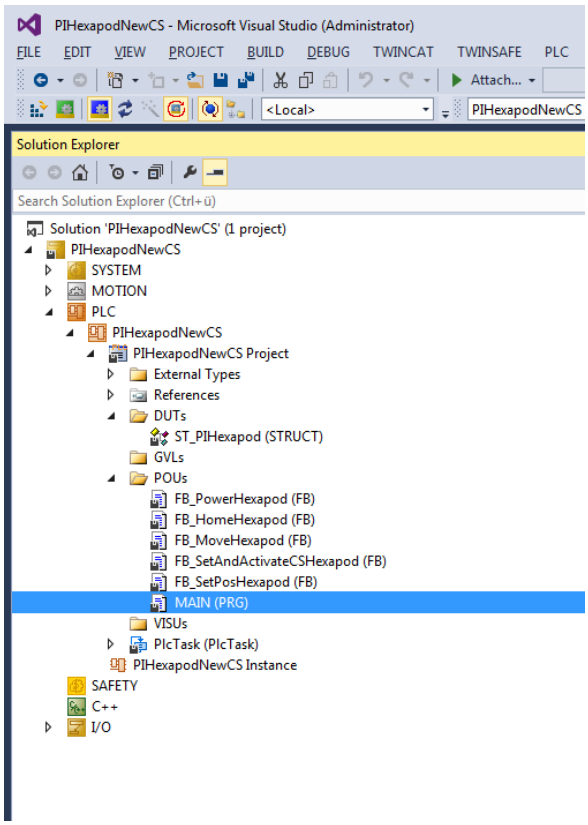
```
       PosW                : LREAL := 0.0;
       bClearPositionLag : BOOL := TRUE;
END_VAR
VAR_OUTPUT
       bDone               : BOOL := FALSE;
       bError              : BOOL := FALSE;
       nErrorId            : UDINT := 0;
END_VAR
VAR
       fbSetPosX           :      MC_SetPosition;
       fbSetPosY           :      MC_SetPosition;
       fbSetPosZ           :      MC_SetPosition;
       fbSetPosU           :      MC_SetPosition;
       fbSetPosV           :      MC_SetPosition;
       fbSetPosW           :      MC_SetPosition;
END_VAR
```

- And copy the following code to the “FB_SetPosHexapod” function block:

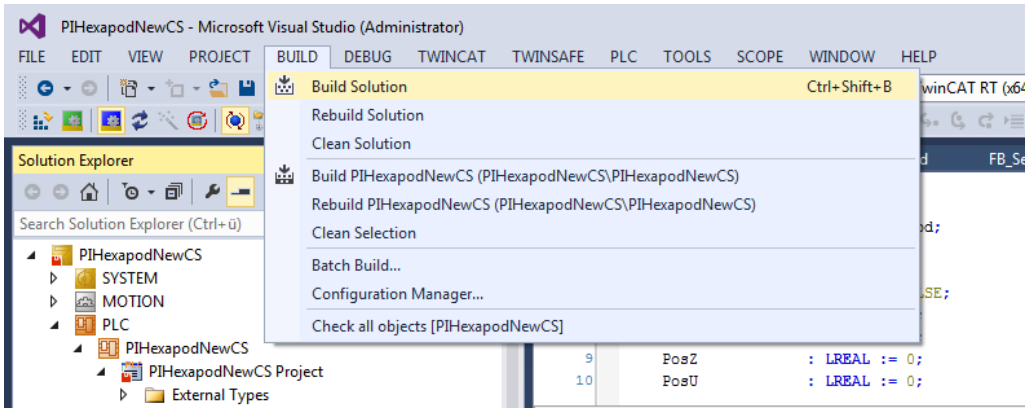
```
(* ToDo: extend error handling *)
(* Set position for all hexapod axes to 0.0 *)

fbSetPosX.Options.ClearPositionLag:= bClearPositionLag;
fbSetPosY.Options.ClearPositionLag:= bClearPositionLag;
fbSetPosZ.Options.ClearPositionLag:= bClearPositionLag;
fbSetPosU.Options.ClearPositionLag:= bClearPositionLag;
fbSetPosV.Options.ClearPositionLag:= bClearPositionLag;
fbSetPosW.Options.ClearPositionLag:= bClearPositionLag;

fbSetPosX(
       Axis:= stPIHexapod.stX ,
       Position:= PosX ,
       Mode:= FALSE,
       Execute:= Execute,
       Busy=> ,
       Done=> ,
       Error=> ,
       ErrorID=> );

fbSetPosY(
       Axis:= stPIHexapod.stY ,
       Position:= PosY ,
       Mode:= FALSE,
       Execute:= Execute,
       Busy=> ,
       Done=> ,
       Error=> ,
       ErrorID=> );
fbSetPosZ(
       Axis:= stPIHexapod.stZ ,
       Position:= PosZ ,
       Mode:= FALSE,
       Execute:= Execute,
       Busy=> ,
       Done=> ,
       Error=> ,
       ErrorID=> );
```

```
fbSetPosU(
       Axis:= stPIHexapod.stU ,
       Position:= PosU ,
       Mode:= FALSE,
       Execute:= Execute,
       Busy=> ,
       Done=> ,
       Error=> ,
       ErrorID=> );
fbSetPosV(
       Axis:= stPIHexapod.stV ,
       Position:= PosV ,
       Mode:= FALSE,
       Execute:= Execute,
       Busy=> ,
       Done=> ,
       Error=> ,
       ErrorID=> );
fbSetPosW(
       Axis:= stPIHexapod.stW ,
       Position:= PosW ,
       Mode:= FALSE,
       Execute:= Execute,
       Busy=> ,
       Done=> ,
       Error=> ,
       ErrorID=> );

IF fbSetPosX.Error THEN
       bError := TRUE;
       nErrorId := fbSetPosX.ErrorID;
ELSE
       IF fbSetPosX.Done AND fbSetPosY.Done AND fbSetPosZ.Done AND fbSetPosU.Done AND
fbSetPosV.Done AND fbSetPosW.Done THEN
              bDone := TRUE;
       ELSE
              bDone := FALSE;
       END_IF
END_IF
```

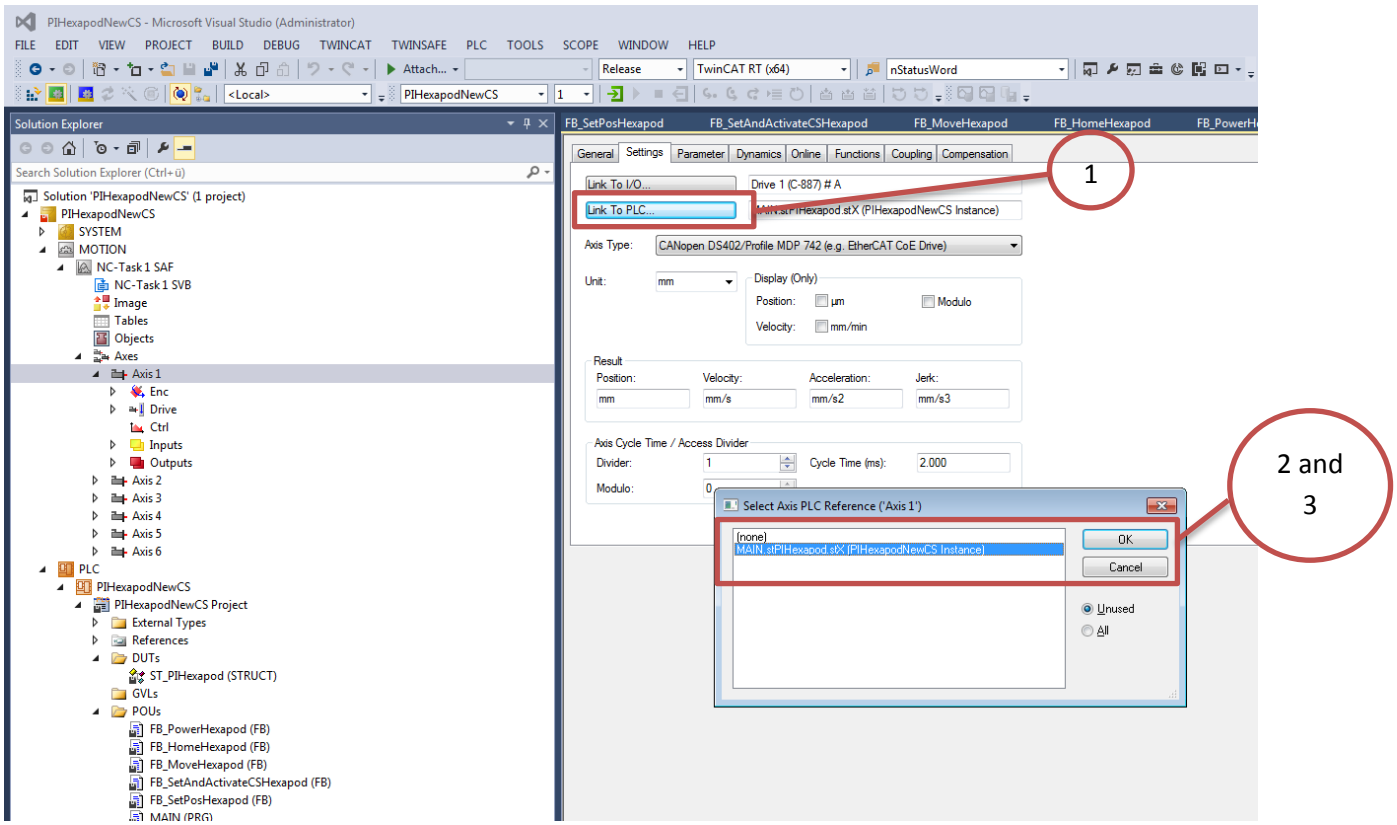
The final PLC structure should look like this:



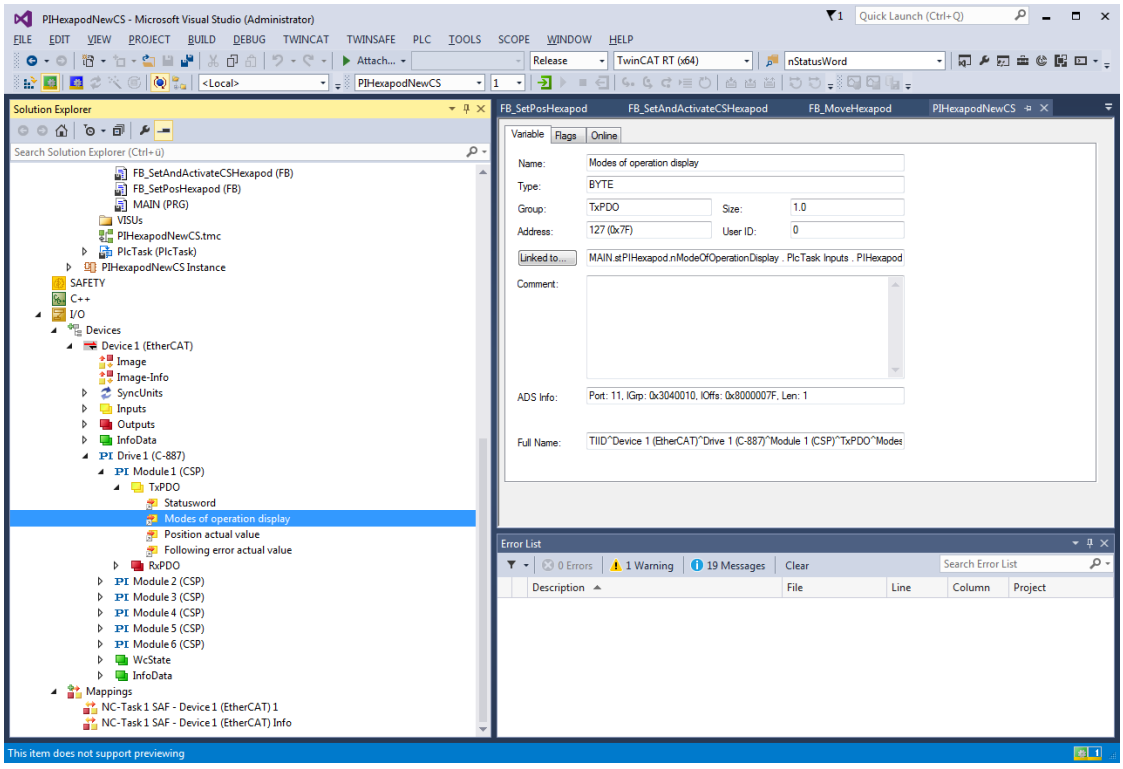
- Build the PLC project once to make the new variables visible (a task configuration will be created automatically).

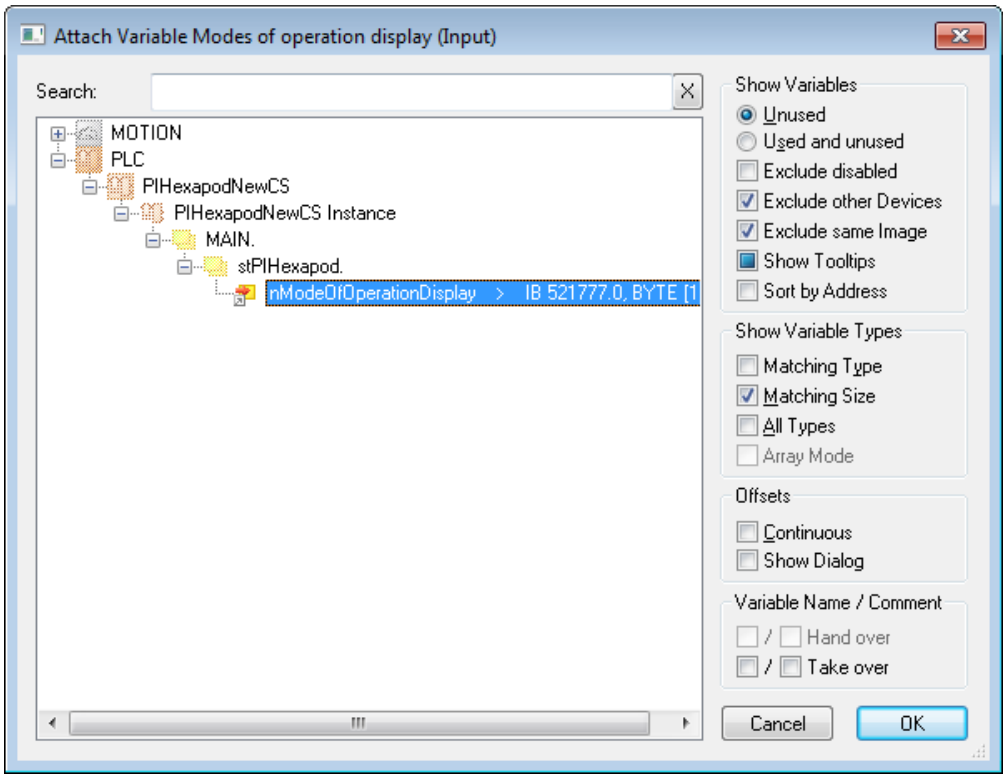


- Link MOTION axis to the corresponding PLC axis reference. Do this for each axis:

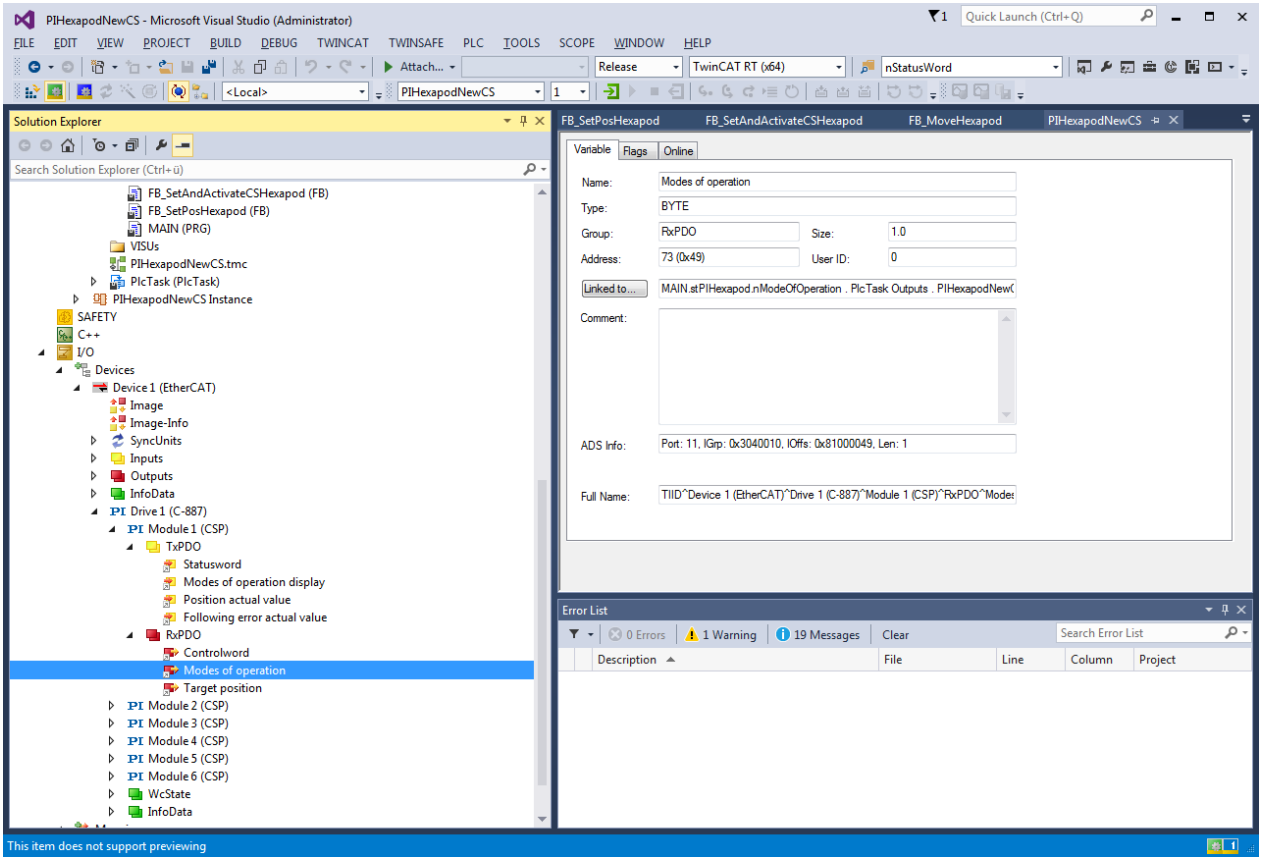


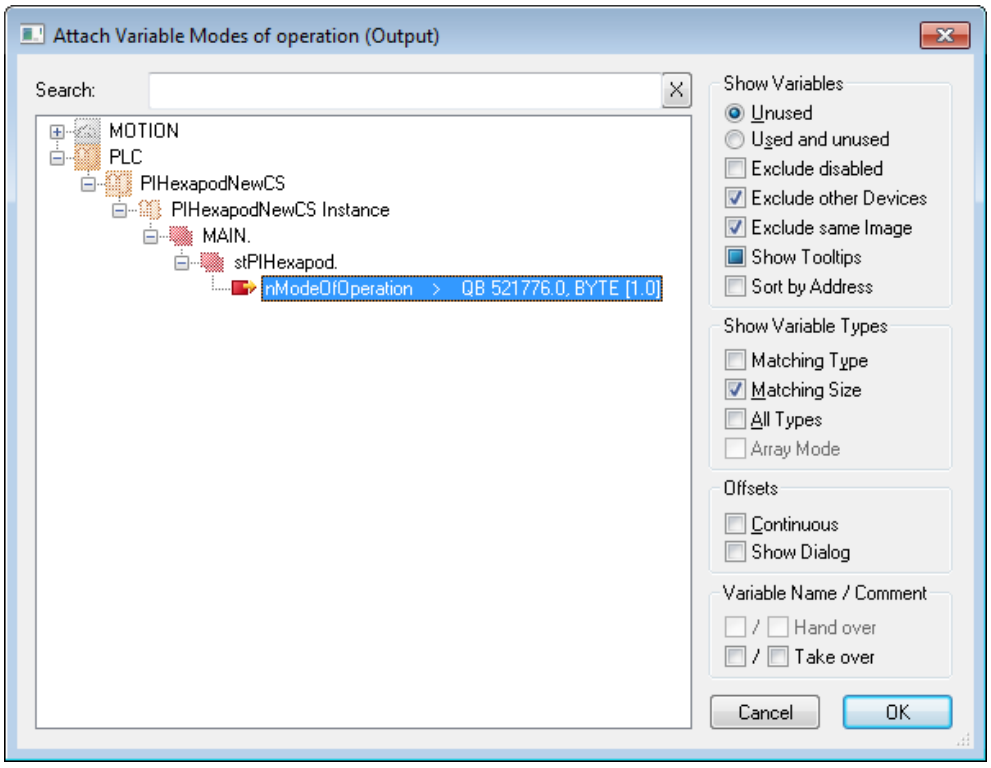
- Link the nModeOfOperationDisplay variable to Modes of operation display of Module 1:
  PLC -> PIHexapodNewCS -> PIHexapodNewCS Instance -> Main.stPIHexapod.nModeOfOperationDisplay




Attach Variable Modes of operation display (Input)
Search:
MOTION
PLC
PIHexapodNewCS
PIHexapodNewCS Instance
MAIN.
stPIHexapod.
nModeOfOperationDisplay > IB 521777.0, BYTE [1
Show Variables
Unused
Used and unused
Exclude disabled
Exclude other Devices
Exclude same Image
Show Tooltips
Sort by Address
Show Variable Types
Matching Type
Matching Size
All Types
Array Mode
Offsets
Continuous
Show Dialog
Variable Name / Comment
/ Hand over
/ Take over
Cancel
OK

- Link the ModeOfOperation variable to Modes of operations of Module 1:
  PLC -> PIHexapodNewCS -> PIHexapodNewCS Instance -> Main.stPIHexapod.nModeOfOperation





- Link the nStatusWord variable (additionally to nState1 and nState2) to Statusword of Module 1:
  PLC -> PIHexapodNewCS -> PIHexapodNewCS Instance -> Main.stPIHexapod. nStatusWord

- Finally the links for the Input variables should look like this:

	Name	Type	Size [Byte]	Linked to
Axis X	Statusword	UINT	2	MAIN.stPIHexapod.nStatusWord, nState1, nState2
	Modes of operation display	BYTE	1	MAIN.stPIHexapod.nModeOfOperationDisplay . PlcTask Inputs . PIHexapodNewCS Instance . PIHexapodNewCS
	Position actual value	DINT	4	nDataIn1 . In . Inputs . Enc . Axis X . Axis X . Axes . NC-Task 1 SAF
	Following error actual value	DINT	4	nDataIn1 . In . Inputs . Drive . Axis X . Axis X . Axes . NC-Task 1 SAF
Axis Y	Statusword	UINT	2	nState1, nState2
	Modes of operation display	BYTE	1	
	Position actual value	DINT	4	nDataIn1 . In . Inputs . Enc . Axis Y . Axis Y . Axes . NC-Task 1 SAF
	Following error actual value	DINT	4	nDataIn1 . In . Inputs . Drive . Axis Y . Axis Y . Axes . NC-Task 1 SAF
Axis Z	Statusword	UINT	2	nState1, nState2
	Modes of operation display	BYTE	1	
	Position actual value	DINT	4	nDataIn1 . In . Inputs . Enc . Axis Z . Axis Z . Axes . NC-Task 1 SAF
	Following error actual value	DINT	4	nDataIn1 . In . Inputs . Drive . Axis Z . Axis Z . Axes . NC-Task 1 SAF
Axis U	Statusword	UINT	2	nState1, nState2
	Modes of operation display	BYTE	1	
	Position actual value	DINT	4	nDataIn1 . In . Inputs . Enc . Axis U . Axis U . Axes . NC-Task 1 SAF
	Following error actual value	DINT	4	nDataIn1 . In . Inputs . Drive . Axis U . Axis U . Axes . NC-Task 1 SAF
Axis V	Statusword	UINT	2	nState1, nState2
	Modes of operation display	BYTE	1	
	Position actual value	DINT	4	nDataIn1 . In . Inputs . Enc . Axis V . Axis V . Axes . NC-Task 1 SAF
	Following error actual value	DINT	4	nDataIn1 . In . Inputs . Drive . Axis V . Axis V . Axes . NC-Task 1 SAF
Axis W	Statusword	UINT	2	nState1, nState2
	Modes of operation display	BYTE	1	
	Position actual value	DINT	4	nDataIn1 . In . Inputs . Enc . Axis W . Axis W . Axes . NC-Task 1 SAF
	Following error actual value	DINT	4	nDataIn1 . In . Inputs . Drive . Axis W . Axis W . Axes . NC-Task 1 SAF
linked automatically by TwinCAT	WcStateOut	BIT	0.1	
	WcStateIn	BIT	0.1	nState4, nState4
	InputToggle	BIT	0.1	nState4, nState4
	State	UINT	2	
	AdsAddr	AMSADDR	8	
	Chn0	USINT	1	
	DcOutputShift	DINT	4	nDcOutputTime . In . Inputs . Drive . Axis 6 . Axis 6 . Axes . NC-Task 1 SAF
	DcInputShift	DINT	4	nDcInputTime . In . Inputs . Enc . Axis 6 . Axis 6 . Axes . NC-Task 1 SAF

- Finally the links for the Output variables should look like this:

Axis X	Controlword	UINT	2	nCtrl1, nCtrl2
	Modes of operation	BYTE	1	MAIN.stPIHexapod.nModeOfOperation . PlcTask Outputs . PIHexapodNewCS Instance . PIHexapodNewCS
	Target position	DINT	4	nDataOut1 . Out . Outputs . Drive . Axis X . Axis X . Axes . NC-Task 1 SAF
Axis Y	Controlword	UINT	2	nCtrl1, nCtrl2
	Modes of operation	BYTE	1	
	Target position	DINT	4	nDataOut1 . Out . Outputs . Drive . Axis Y . Axis Y . Axes . NC-Task 1 SAF
Axis Z	Controlword	UINT	2	nCtrl1, nCtrl2
	Modes of operation	BYTE	1	
	Target position	DINT	4	nDataOut1 . Out . Outputs . Drive . Axis Z . Axis Z . Axes . NC-Task 1 SAF
Axis U	Controlword	UINT	2	nCtrl1, nCtrl2
	Modes of operation	BYTE	1	
	Target position	DINT	4	nDataOut1 . Out . Outputs . Drive . Axis U . Axis U . Axes . NC-Task 1 SAF
Axis V	Controlword	UINT	2	nCtrl1, nCtrl2
	Modes of operation	BYTE	1	
	Target position	DINT	4	nDataOut1 . Out . Outputs . Drive . Axis V . Axis V . Axes . NC-Task 1 SAF
Axis W	Controlword	UINT	2	nCtrl1, nCtrl2
	Modes of operation	BYTE	1	
	Target position	DINT	4	nDataOut1 . Out . Outputs . Drive . Axis W . Axis W . Axes . NC-Task 1 SAF