Operating a FAULHABER CO driver out of a CODESYS environment

Summary

FAULHABER MotionController or MotionControlSystems having the CANopen interface can be operated from a CODESYS environment. Here the operation as a SoftMotionLight axis is demonstrated. A SoftMotion-Light (SML) axis uses the profile based operating modes PP and PV of the drive where the trajectory of the movement is calculated within the drive, reducing the requirements towards the communication update rate. Therefore, using a SML rather than a SoftMotion axis might be preferred in a CAN environment.

The description includes how to add CAN connectivity to the used RaspberryPi environment and how to adapt the .eds files of the drives to be visible in the CODESYS environment.

Applies To

Any FAULHABER MotionController and MotionControlSystems having a CANopen interface

MC V2.5

MCxx 300x F/P/S CO

22xx BX4 COD, 32xx BX4 CO

MC V3.0

MC 5004 P RS/CO

MC 5005 S CO, MC 5010 S CO

MCS32xxBX4 CO, MCS 3274BP4 CO

Description

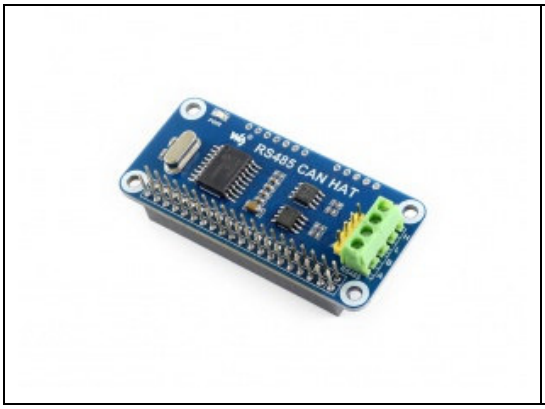
In this example a RaspberyPi is used as the target of the CODESYS environment. Out of the box there is no CAN interface at a RaspberyPi, it can be added by means of a plug-in board, a hat. The necessary drivers can easily be added as that's often the case in the RaspberryPi environment.

When this was successful CODESYS needs to be aware of the CAN connectivity too which proved to be the most challenging part here Prerequesites to configure the MC.

If you don't have to deal with the RaspberryPi simply jump over these sections and start directly with the.

General information about CAN and CANopen system can be found in our Product Application Note 174.

Add CAN connectivity to a RaspberyPi environment

	The RaspberryPi comes without CAN connectivity. There are however different plug-in modules which can fix that. They usually use one or two MICROCHIP® CAN controllers like the MCP 2515 which are connected to the BCM on the Raspberry via SPI. Here I used a Waveshare RS485 CAN hat More information can be found at: https://www.waveshare.com/wiki/RS485_CAN_HAT

Driver wise there are two steps.

Active the CAN low level driver

To activate the low-level driver for the hat you need to add one or two lines to the /boot/config.txt file:

```
sudo nano /boot/config.txt
```

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25,spimaxfrequency=2000000
dtoverly=spi_bcm2835-overlay
```

These changes of course need to be saved and a restart is required. As for the mcp2515 line it depends on what crystal is used in the hat. The one used here had a 12 MHz crystal but there seem to be older ones having an 8 MHz crystal.

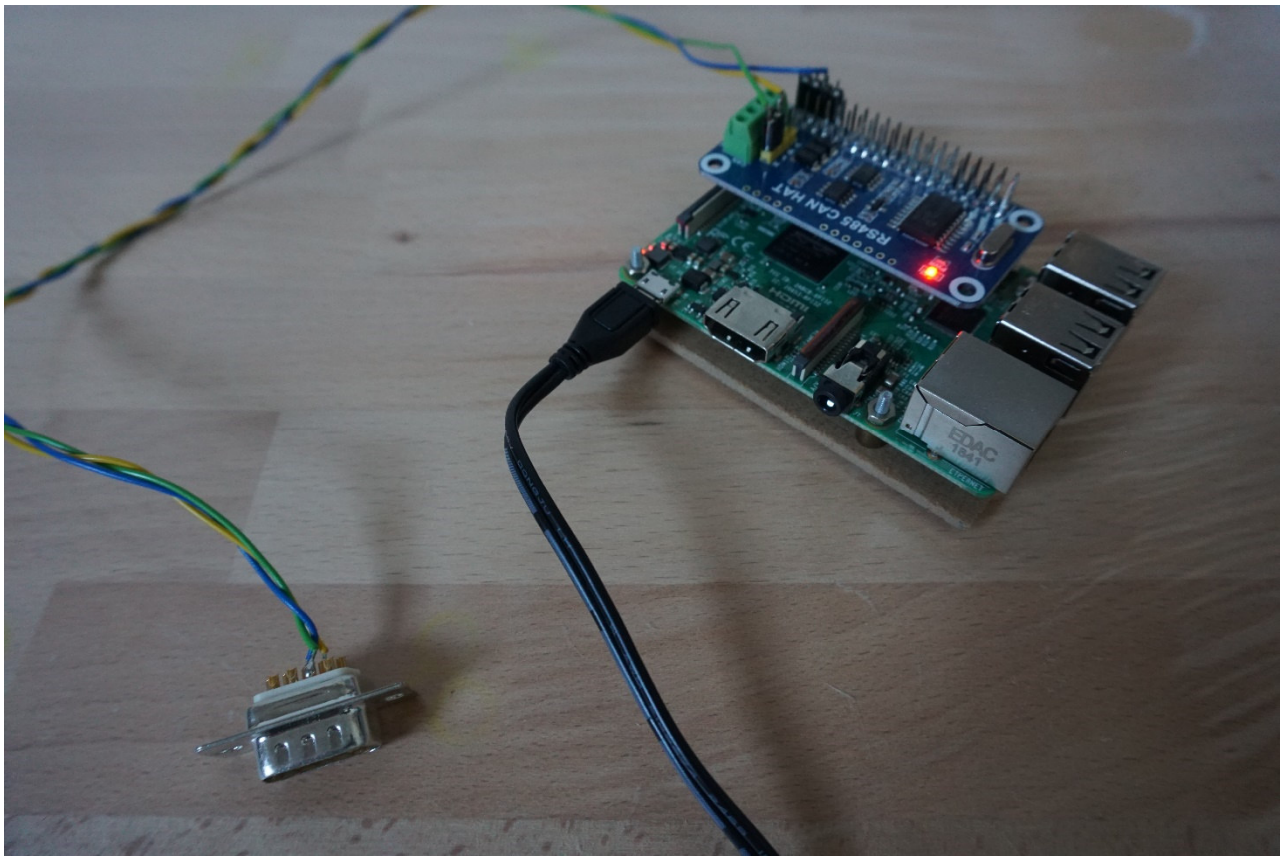


Figure 1 RaspberryPi 3B equipped with the RS485 CAN HAT and a CAN cable

In the end your setup might loop like Figure 1. Please note CAN is a 3-wire interface. It's no current but a voltage interface which requires a common reference. Therefore, a GND-line here had to be added in addition to the original 2 terminals (CAN_H/CAN_L) at the hat.

Add the CAN tools

Next step is to add the high-level drivers by

```
sudo apt update
sudo apt install can-utils
```

After a reboot you should be able to see the CAN0 in your file system:

```
ls /sys/bus/spi/devices/spi0.0/net
can0
```

and

```
ls /sys/bus/spi/devices/spi0.0/net/can0
```

would list additional settings of this driver.

The driver interface of the CAN is one of a net type. So, it should show up in `ip addr` but will initially be down.

There is online documentation available for the RaspberryPi on how to permanently enable the CAN interface and of course how-to low-level test the CAN communication, which requires a second CAN device of course. Here the CODESYS environment will enable the CAN communication. That way the CAN baud-rate can be configured by the CODESYS run time environment depending on the settings within the project.

Have CODESYS use the CAN connectivity

CODESYS can start the CAN with the correct baud rate. So, no need to have CAN activated by the Raspian all the time.

Stating from the run-time revision 3.5.12.20 the script `rts_set_baud.sh` is expected to be found at in `/var/opt/codesys/`

We can create it using nano:

```
sudo nano /var/opt/codesys/rts_set_baud.sh
```

and add:

```
#!/bin/sh
BITRATE=`expr $2 \\* 1000`
ifconfig $1 down
echo ip link set $1 type can bitrate $BITRATE
ip link set $1 type can bitrate $BITRATE
ifconfig $1 up
```

Challenge was having the correct commas around the definition of the variable BITRATE.

After creating the file, we need to have it executable and therefore call:

```
sudo chmod +x /var/opt/codesys/rts_set_baud.sh
```

Afterwards, after calling the script manually:

```
/var/opt/codesys/rts_set_baud.sh can0 250
```

A call of `ip addr` should end up with an activated CAN0. We should then again be able to use the low-level CAN handlers using calls of `cansend` or `candump` from a terminal.

Older versions of the runtime seem to expect the very same script at `/root/rts_set_baud.sh`.

Prerequisites – configure the MC

Before starting to set-up your CODESYS environment please use the MotionManager to configure your drive:

- Select the motor and feedback-system by using the “Select motor” tool of the MotionManager
- Configure the load inertia by using the “Configure controller” tool of the MotionManager

- Tune the control-loops and verify the dynamic parameters like acceleration and deceleration which can be safely used in your application by operating the drive using the MotionManager “Tuning window”
- Configure whatever additional I/Os might be used e.g. for a drive-based homing sequence using the MotionManager “Drive functions” window.
- Save your settings in the drive and store a copy of it at your PC

CODESYS project configuration

The general setup is the same as detailed in AppNote 164:

- Create a new project
- Select the correct target environment – here it is a RaspberryPi
- Connect to the target device

Setting up a CAN environment

After the first generic steps we now add the CAN connectivity. Steps are:

- Add a CAN-bus to your project and check the Network Id and the Baudrate, which is specific for this project (Figure 2).

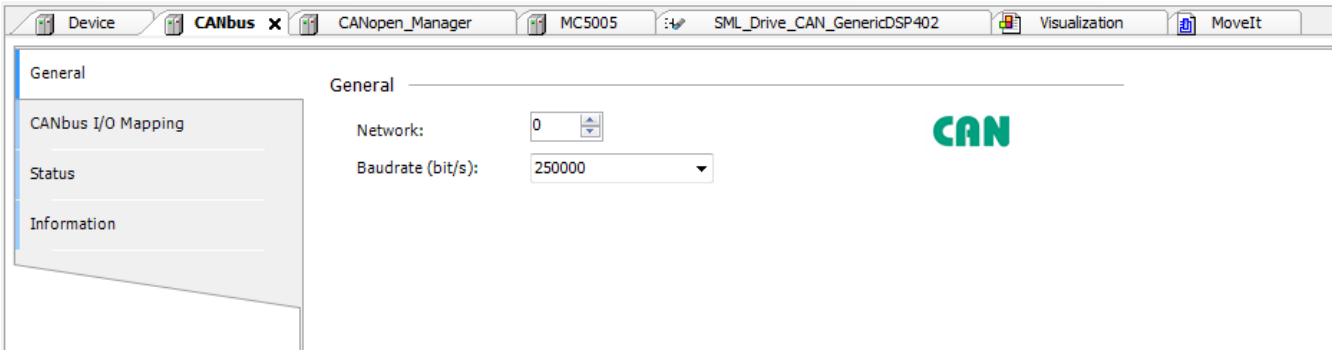


Figure 2 CAN-bus specific settings

These are the parameters used to call the `rts_set_baud.sh` script.

- Add a CAN Manager below the CAN-bus and configure the services which are:
  - Start the slaves
  - Use either the guarding or the heartbeat service to supervise the drives.
  - Configure the cyclic data exchange via the SYNCH interval. An interval length of ~10ms would be a good start for up to 4 nodes. In Figure 3 an update rate of 20ms was configured.

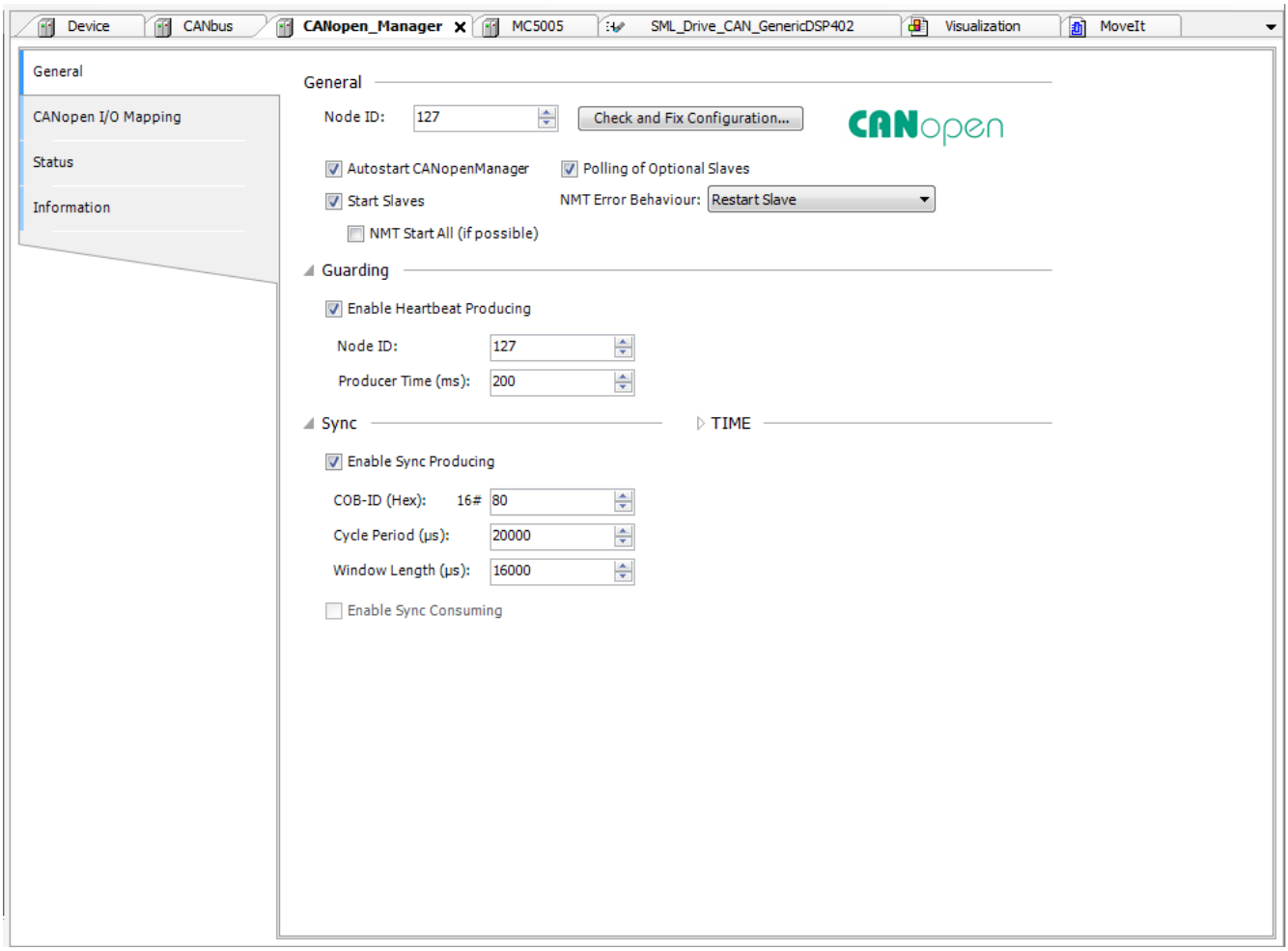


Figure 3 Configuration of the services to be used by the CANopen Manager

- The FAULHABER MC will then be added as a subsystem to the CANopen master using its context menu but we might have to modify the .eds files first.
- Start thinking about the process-image, which is the collection of parameters to be exchanged cyclically when the system is running.

Prepare the .eds files for usage in a SoftMotion environment

Any CANopen drive can be added to the project using the context menu of the CANopen Manager. Now in order to get a selection of FAULHABER drives in such a case, these drives need to be added to the device database of your CODESYS installation.

This can be done using Device Repository window out of the Tools menu

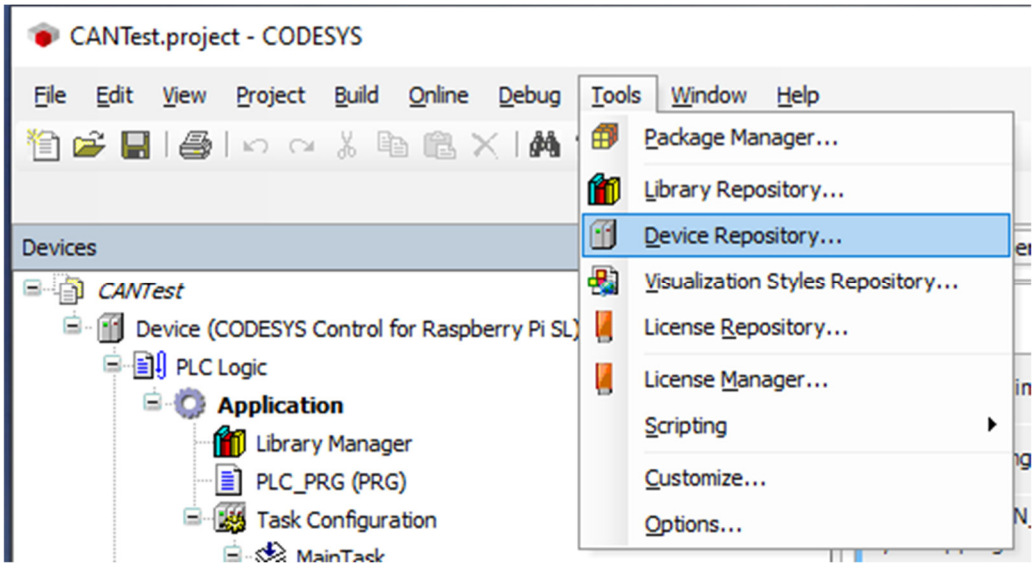


Figure 4 call the CODESYS Device Repository

New devices can be added using the Install button. The electronic device description .eds is needed. All .eds of FAULHABER drives can be found in the program directory where the MotionManager has been installed in the sub-folder EDS.

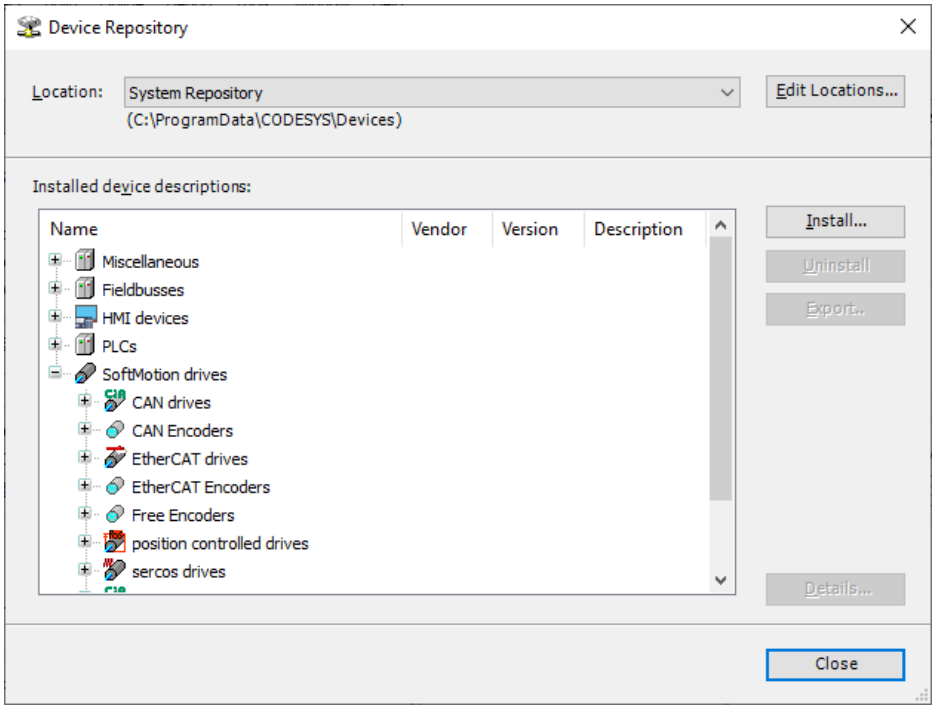


Figure 5 CODESYS Device Repository window

When selecting the .eds files and importing them for firmware revisions up to Rev K you will only get a single FAULHABER MotionController displayed afterwards.

We need to modify the ones we want to have by changing some entries. Editing can be done using any text editor – it's an ini file format.
If your .eds is up to revision K it needs to be modified. Starting with rev L everything should be fine.

Table 1 Overview of the .eds files and the connected products

File name	Applies for driver	Suggested ProductNumber entry for the .eds file
605.0101.01_x.eds	MC 5010 S CO where _x denotes the firmware revision	010101yy where yy is the revision: 10 for J, 11 for K, 12 for L, …
605.0111.01_x.eds	MCS32xx CO where _x denotes the firmware revision	011101yy where yy is the revision: 10 for J, 11 for K, 12 for L, …
605.0121.01_x.eds	MC 5004 P RS/CO where _x denotes the firmware revision	012101yy where yy is the revision: 10 for J, 11 for K, 12 for L, …
605.0121.03_x.eds	MC 5004 P RS/CO STO where _x denotes the firmware revision	012103yy where yy is the revision: 10 for J, 11 for K, 12 for L, …
605.0131.01_x.eds	MC 5005 S CO where _x denotes the firmware revision	013101yy where yy is the revision: 10 for J, 11 for K, 12 for L, …
605.3150.67_x.eds	MCBL V2.5 CANopen drives where _x denotes the firmware revision	315067yy where yy is the revision: 02 for B, 03 for C, 04 for D, …
605.3150.68_x.eds	MCDC V2.5 CANopen drives where _x denotes the firmware revision	315068yy where yy is the revision: 02 for B, 03 for C, 04 for D, …
605.3150.69_x.eds	MCLM V2.5 CANopen drives where _x denotes the firmware revision	315069yy where yy is the revision: 02 for B, 03 for C, 04 for D, …
605.3150.70_x.eds	MCBL AES V2.5 CANopen drives where _x denotes the firmware revision	315070yy where yy is the revision: 02 for B, 03 for C, 04 for D, …

There are a few modifications:

1. Add the default value for the Device Type (0x1000) to indicate a CiA 402 servo-drive

```
[1000]
ParameterName=Device type
ObjectType=7
DataType=0x0007
AccessType=RO
DefaultValue=0x00420192
```

```
PDOMapping=0
```

2. Change the FileRevision to a number reflecting the FW revision. So for a Rev K use 11.

```
[FileInfo]
CreatedBy=Dr. Fritz Faulhaber GmbH & Co KG
ModifiedBy=Dr. Fritz Faulhaber GmbH & Co KG
Description=MC3
CreationTime=09:34PM
CreationDate=10-25-2019
ModificationTime=09:34PM
ModificationDate=10-25-2019
FileName=605.0101.01-K.eds
FileVersion=1
FileRevision=11
EDSVersion=4.0
```

3. Modify the ProductNumber listed in the [DeviceInfo] section to be unique for this specific firmware revision and device. Use the entry out of Table 1.

```
[DeviceInfo]
VendorName=Faulhaber
VendorNumber=0x147
ProductName=MCS *
ProductNumber=01110111
RevisionNumber=0x00010000
```

4. Add parts of information in the Identity object (0x1018):
    a. Add the default value for the enty vendor Id

```
[1018sub1]
ParameterName=Vendor ID
ObjectType=7
DataType=0x0007
AccessType=RO
DefaultValue=0x00000147
PDOMapping=0
```

5. Save the modifications and only now import the interesting ones to CODESYS.

Add a FAULHABER MC to the project

After having the modified .eds files imported to the Device Repository a right click on the CANopen Manager should offer the selection of drives.

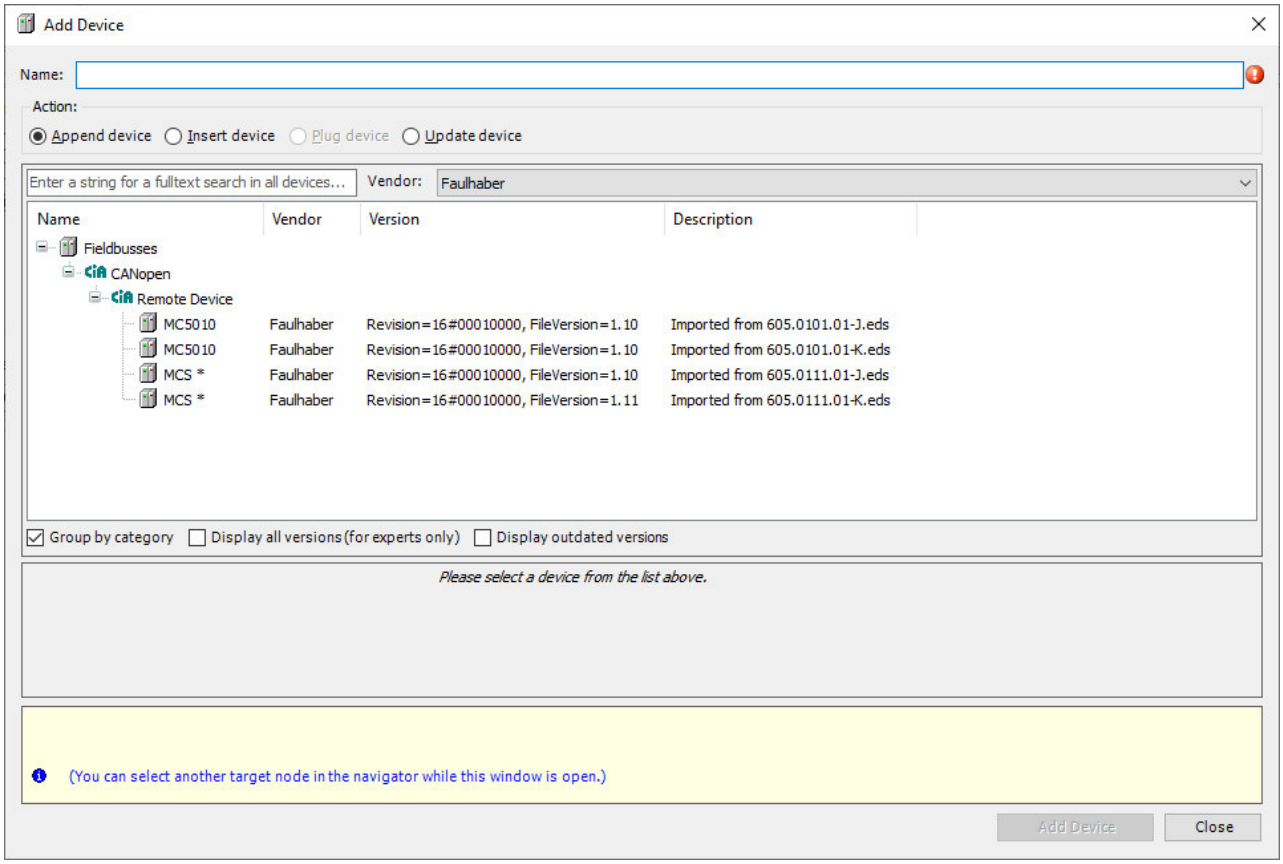


Figure 6 Device Selection for the CAN-node to be added - here having a Filter to FAULHABER being the supplier

If your driver is found, use it, otherwise re-check the modifications in the .eds file.

After adding whatever number of drives shall be part of the project, their settings must be checked, and the process image needs to be configured.

Communication related settings

Within the “General” tab of the drives settings we might enable the Expert Settings which will allow access

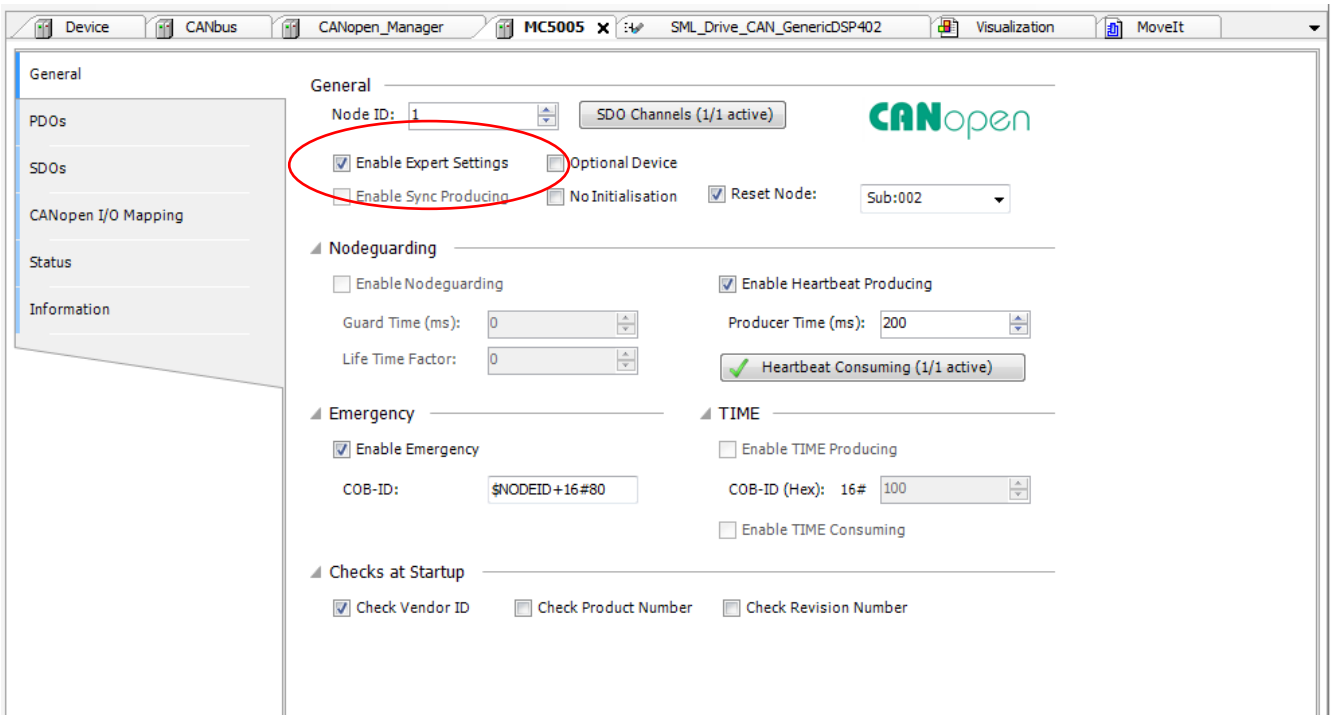


Figure 7 General settings of the inserted CAN drive

to all the services to be configured.

Main setting here would be the node ID of the drive of course plus whether the drive shall be treated as an optional one and which type of Nodeguarding to be used.

The classic nodeguaring seems to be a little more robust. A Guard Time is to be defined after which the master will request a status-update from the drive. If there is no answer – which happens under heavy communication load – the Life Time factor determines the number of tries until the drive is considered to be lost and an error can be generated. A typical setting could be a Guard Time of 100ms and a Life Time factor of 3, but that of course depends of the safety requirements of the application.

To active Nodeguarding, disable Heartbeat first and then enable Nodeguarding.

If Heartbeat is to be used the producer Time must be shorter than the respective consumer time. So, the producer time of the drive to be supervised should be shorter than the consumer time which is the limit for a heartbeat message to be received. Unless these settings can be handled separately give nodeguaring a try.

Add a SoftMotionLight axis to the drive

Next, we could plug a SoftMotionLight CiA402 axis into the device node using its context menu.

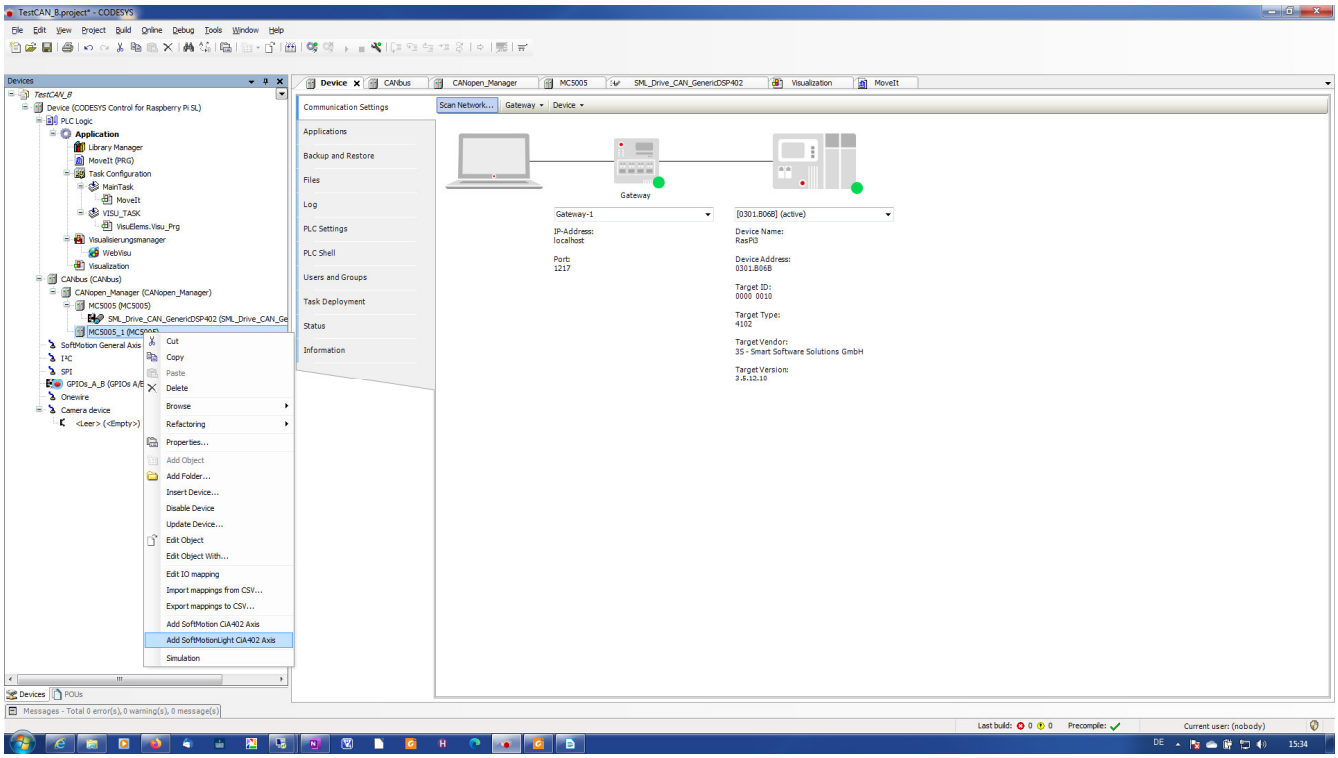


Figure 8 Add a SoftMotionLight axis to the drive

Two steps would be necessary to be able to finally use it:

a) Align the scaling of the movement between the SoftMotionLight axis and the MotionController
b) Create an appropriate PDO mapping for the CAN node and link the resulting process image to the SoftMotionLight axis.

Adjust the settings of the SoftMotionLight axis

Scaling of the SoftMotion axis

For each SML drive there is one combined settings page where the scalings of the motion as to be used in the PLC and the mapping of the parameters exchanged with the device are configured (Figure 11). The entries in Figure 11 will rescale the drives actual position / target position into the position units used in the application. The entries won't rescale the scaling of speed and acceleration expected by the axis, however.

As for the scaling the base information is the encoder resolution which is used by the drive. So how many increments will the drives actual position move when the motor turned once. This might be the native resolution of the encoder used in this axis but might also have been changed, if the drives Factor Group has been modified.

In combination with the SoftMotionLight axis we recommend leaving the Factor Group of the driver untouched. In such a case the settings here could be:

Increments per motor turn: 4096 inc / turn for a 12-bit encoder

Motor turns / gear output turns: please enter whatever reduction is to be considered. Please use the real reduction as e.g. 63 motor turns per 17 gear output shaft turns on case of a 4:1 22F gearhead.

Gear output turns / units in application: depends on whatever scaling is suitable. If a ball-screw having a pith of 1.0 mm is used and the position shall be handled in µm the entry might be 1/1000. If the position is to be used in ° use 1/360. If the native resolution of the encoder is to be used 1/4096 would be the setting for the 12-bit encoder.

Scaling of the drive velocity and acceleration/deceleration

The scalings in Figure 11 uses the drive position in increments and rescales it to whatever application specific scaling shall be used. There is no such explicit way to rescale the speed of the drive. The SoftMotionLight Axis expects these parameters to be in either [incr/s] or [incr/s²]. The FAULHABER MotionControllers however expect the speed to be in rpm for rotating motors or mm/s for linear motors and the acceleration/deceleration in 1/s². Scaling for the speed could be changed using the Factor Group but the scaling for the acceleration and deceleration are fixed.

What we can do, is adapt factors used by CODESYS to calculate the real values sent over the bus. There are two factors which are to be set:

- **fConstVelFactor** is used internally to compute the scaling in between the inputs of the MC_xxx function blocks and the speed sent to the drive
- **fConstAccFactor** in used to do the same for the acceleration and deceleration values.

These internal factors are hidden away by default but can be accessed for each SML axis if "show generic device configuration views" is activated for the Device editor. The setting can be changed using the Tools/Options menu of the engineering environment (Figure 9).

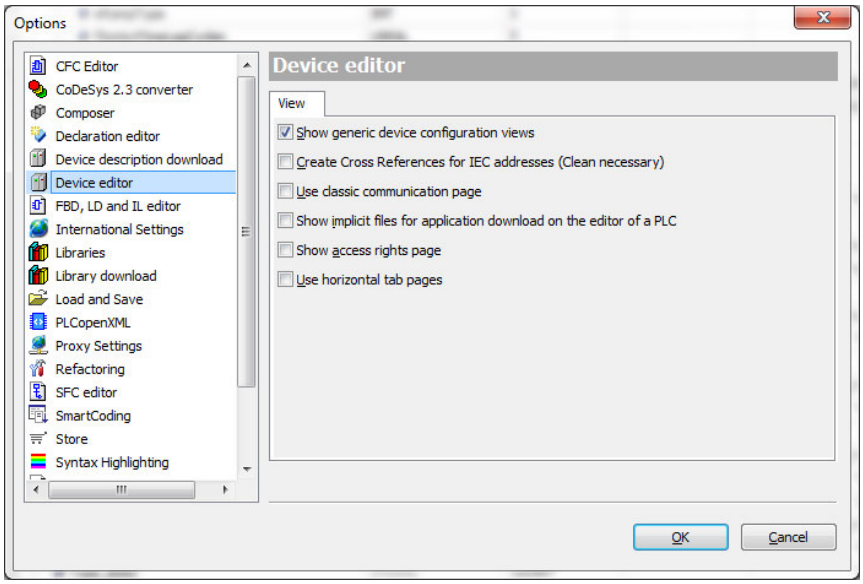


Figure 9 Enable the generic device settings view for the Device editor

The values to be entered are:

- **fConstVelFactor** 60/Encoder resolution e.g. 60/4096
- **fConstAccFactor** 1/Encoder resolution e.g. 1/4096

These entries are expected to be floating numbers. A few digits should be enough as respective actual values might not be that precise either. The acceleration is based on a current measurement with limited precision and even the exact value of the speed might not be that important, unless speed mode is sued to drive at a very exact speed.

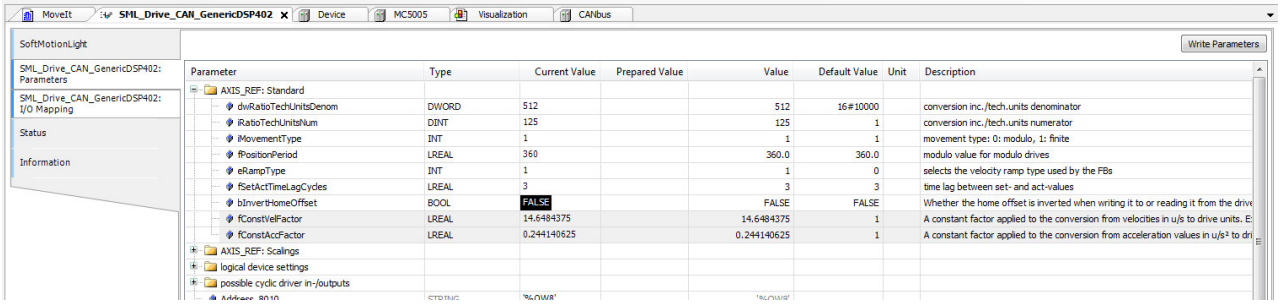


Figure 10 Modified factors to rescale velocity and acceleration

These settings must be configured for each instance of a SoftMotionLight axis in the project.

PDO Mapping

After the scaling, it is the Mapping of the process-image that must be considered. The Scaling/Mapping tab of the SoftMotionLight drive (Figure 11) gives an overview which parameters of a CiA 402 CANopen servo-drive would be used, if available. Plus, there is a checkbox to facilitate automated mapping. This mapping here applies for the mapping between the PDOs configured for the drive and the connected SoftMotionLight axis only. Automated will not generate the PDOs but it will make use of whatever the PDOs are offering.

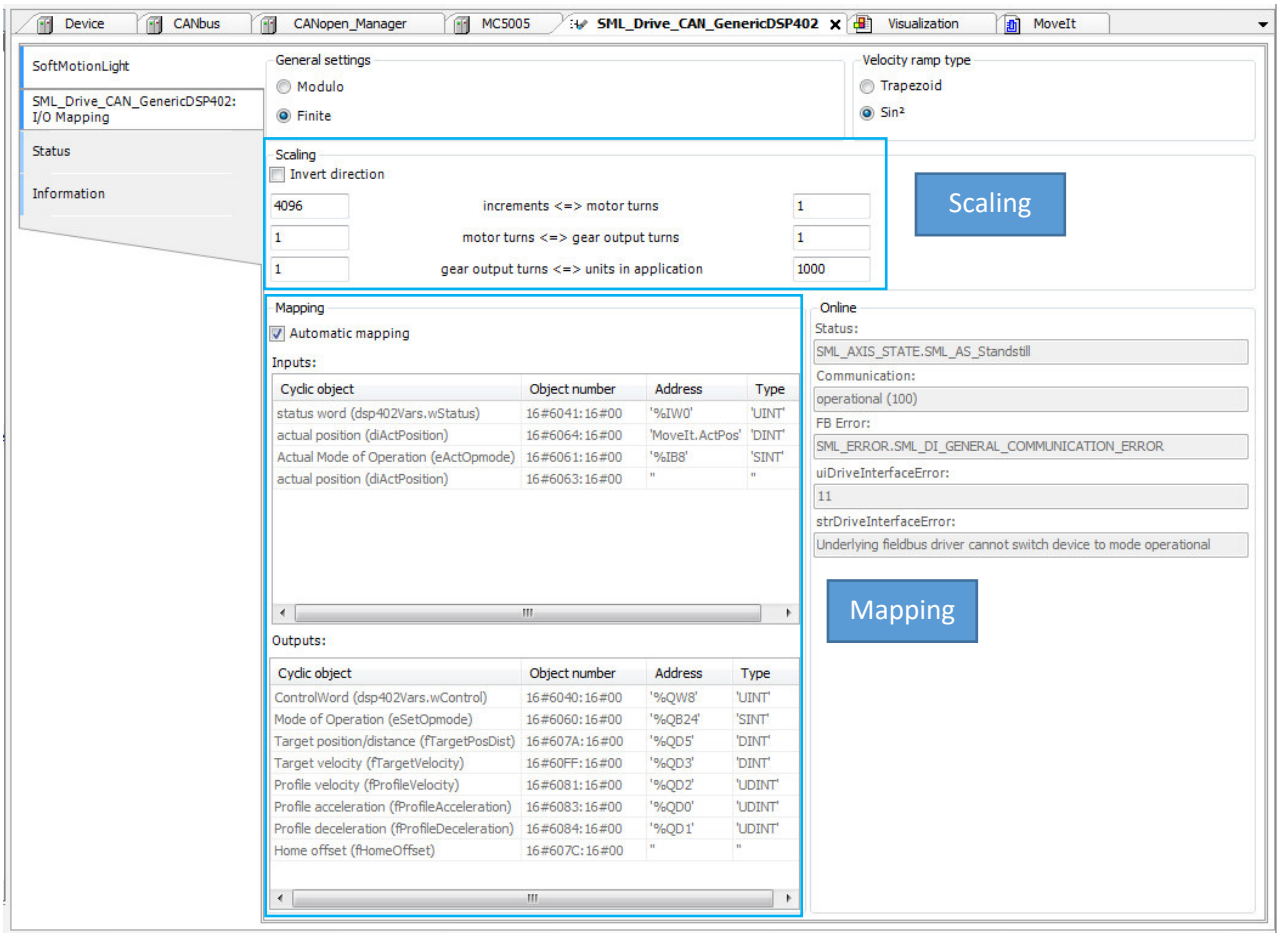


Figure 11 Scalings and mappings of the axis

So, what we must do is, try to make the necessary parameters available.
We do this, by creating an appropriate PDO-mapping for the CANopen drive used here and then un-check and re-check the automatic mapping checkbox.

General information about PDOs and their settings can be found in our Product Application Note 174.

First edit the PDO mapping itself.

The set of parameters which could be used by the SoftMotionLight axis can be reviewed in the mapping part of Figure 11.

The PDOs can be edited from the PDO tab of the device. For each of the PDOs the objects to be transferred can be modified using the + Add Mapping or the x Delete. A double click on any PDO will open a dialog to edit the general communication settings of the PDO. Suggestion would be a mapping according to Figure 12 and Table 2.

Please send the PDO containing the profile parmaeters first, so they are available when sending the Control word, the Target Position and the Modes of operation.

Table 2 Summary of the objects mapped to the process image

From Master to drive (RxPDO)		From drive to Master (TxPDO)	
0x6083.00	Profile Acceleration	**0x6041.00**	Status word
0x6084.00	Profile Deceleration	**0x6064.00**	Position actual value
0x6081.00	Profile Velocity	**0x6061.00**	Modes of operation display
0x60FF.00	Target Velocity		
0x6040.00	Control word		
0x6060.00	Modes of operation		
0x607A.00	Target position		

Additional parameters can of course be mapped depending on the application.

Next check the transmission types of the PDOs. Suggestion is to use the default asynch one for the ones received by the drive (RxPDO) – Transmission Type = 255 - and change the ones transmitted to a cyclic every cycle manner (Figure 13) – Transmission Type = 1.

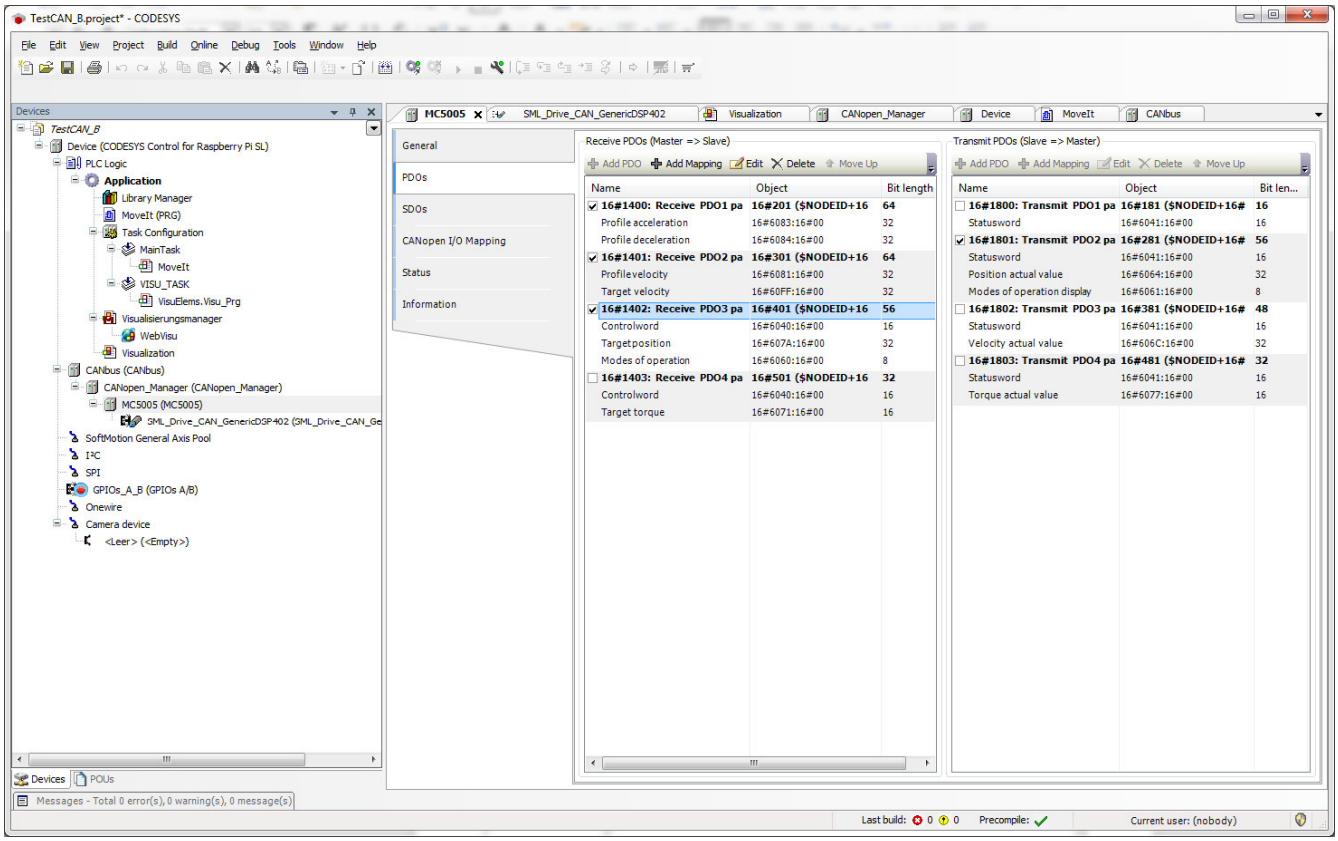


Figure 12 PDO tab of the settings summarizing the PDO mappings

The cycle-time of the communication itself has already been configured as the Synch Period at the CANopen Manager (Figure 3).

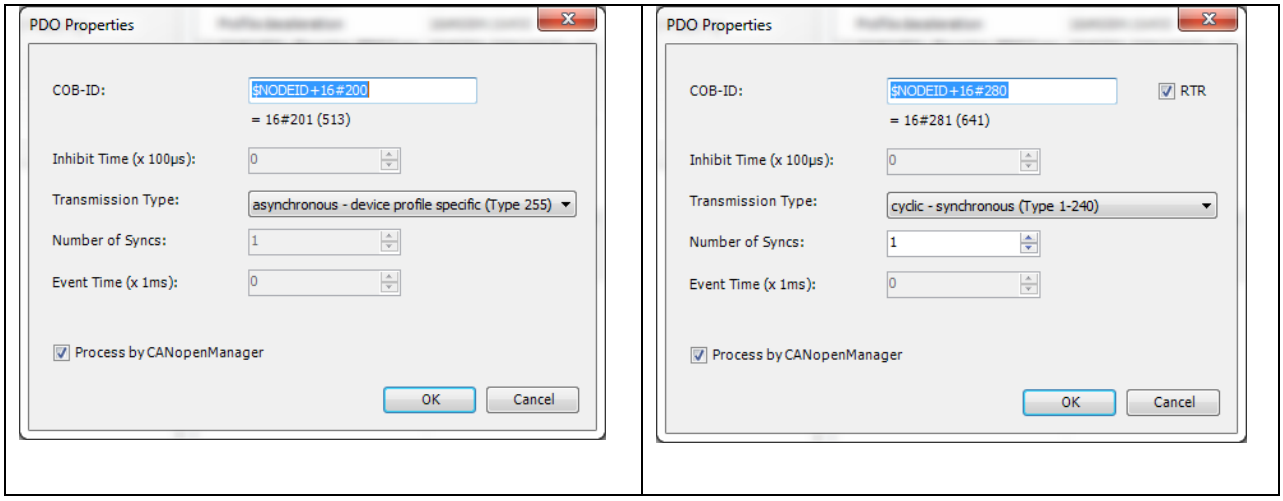


Figure 13 General PDO settings available via Edit or double click

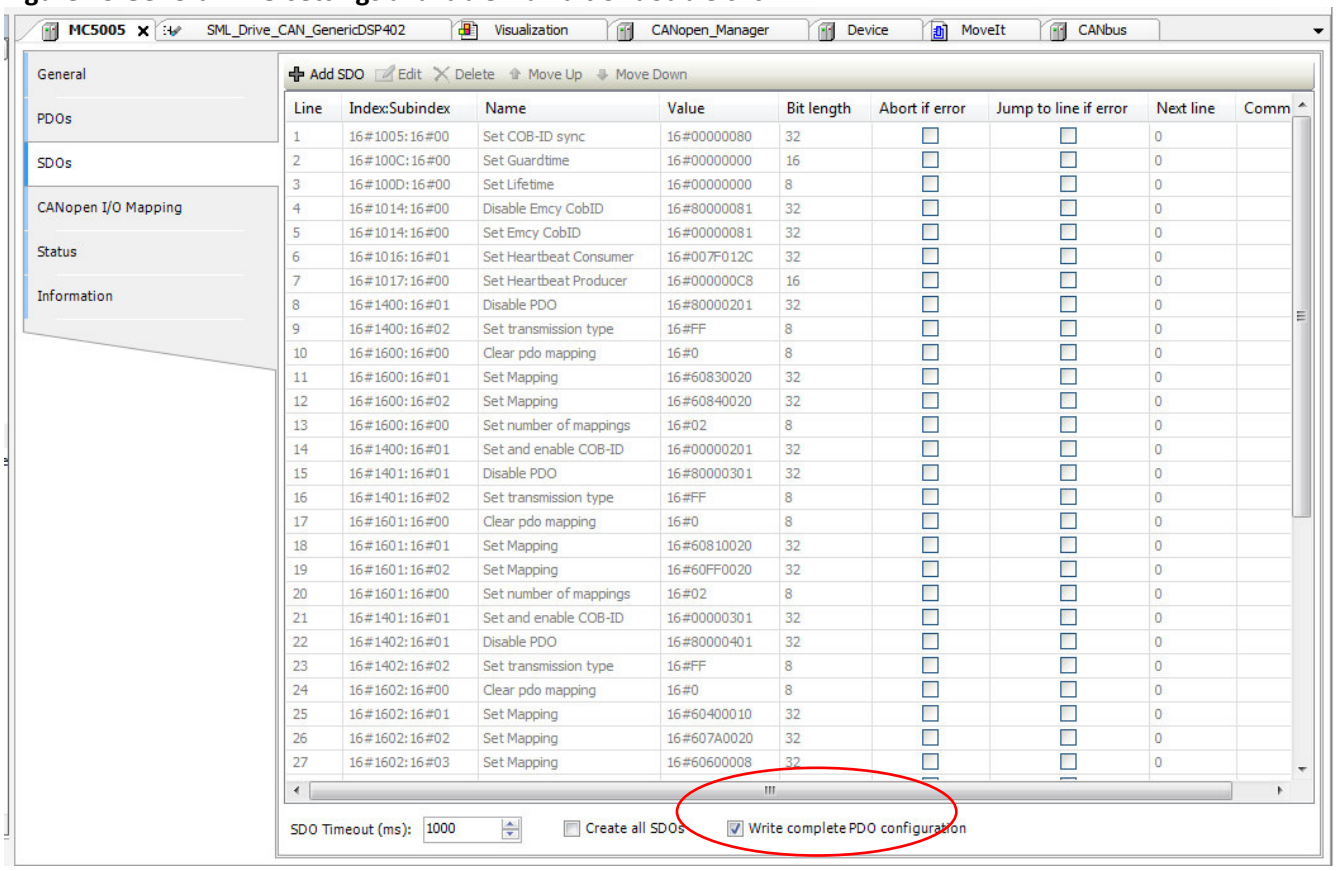


Figure 14 SDO tab of the settings which lets you define which part of the drive configuration shall be updated at startup

Finally make sure to have the checkboxes to download the PDO Assignment and the PDO configuration activated in the SDO tab of the settings (see Figure 14).

Then un-check and -re-check the Automatic mapping checkbox in Figure 11 to have CODESYS re-check the available parameters.

To have CODESYS re-check whether there are new useful parameters for a Soft-Motion axis, un-check and re-check the automatic mapping checkbox in Figure 11.

So whenever you added parameters to the mapping try to add them to the Soft-Motion axis.

Create your application

In our Product Application Note 164, Codesys and FAULHABER V3.0 EtherCAT, the complete application and the interaction with the drive was implemented in structured text. When using a SoftMotionLight axis the basic SoftMotion function blocks can be used to interact with the drive. Like in our Product Application Note 185, where the SoftMotion axis has been used.

There is a more detailed description of the FBs provided with the SoftMotionLight library within the CODESYS online help system.

The SoftMotionLight FBs all use the drive data structure AXIS_REF_CAN_DS402_SML. Each SoftMotion axis creates one instance of it. This structure holds all the necessary data which are the ones exchanged with the drive plus maybe some internal ones too which we don't care about.

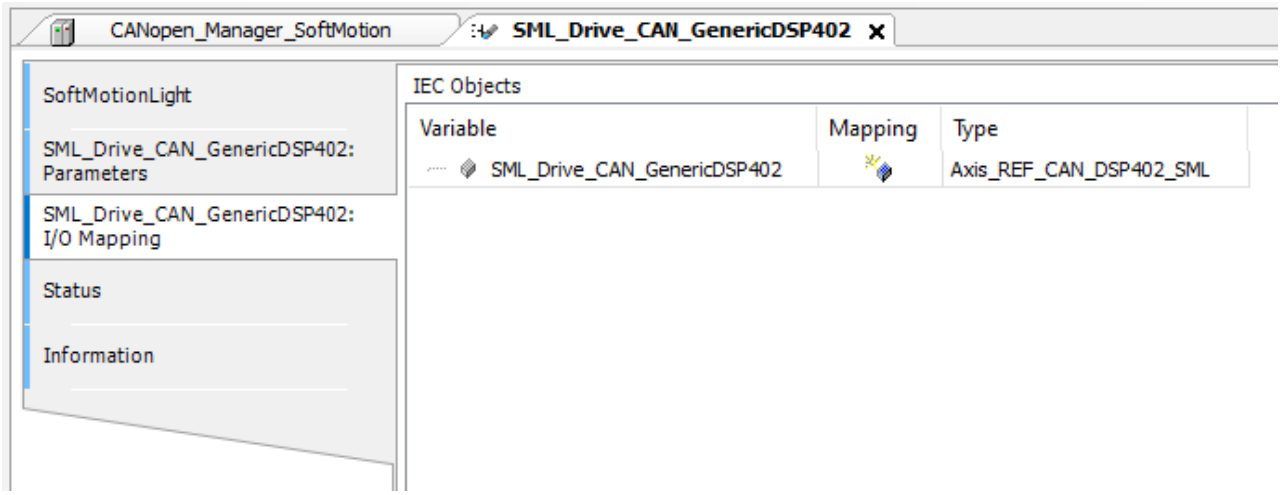


Figure 15 Mapping of the SML axis - this object instance would be SML_Drive_CAN_GenericDSP402

Here a short example using only a few of these FBs was implemented (Figure 17). There is only one real automatic action implemented – moving back to the startpos after a first move. Otherwise the inputs to enable the power stage, so to start the homing and to start a move are simply mapped to a visualization and are then switched manually. The used FBs were:

MC_Reset_SML

The Reset FB can be used to acknowledge a drive error state indicated by bit 3 in the CiA402 status word. MC_Reset_SML must be used to force the transition of the drive's state-machine back into the switch-on disabled state.

The FB must be connected to the correct instance of the axis structure.

The detailed configuration of the drive behavior in case of any detected error can be configured using the MotionManager/ Drive functions / error handling. The drive will end up in FAULT state for any errors where the FAULT mask has been set.

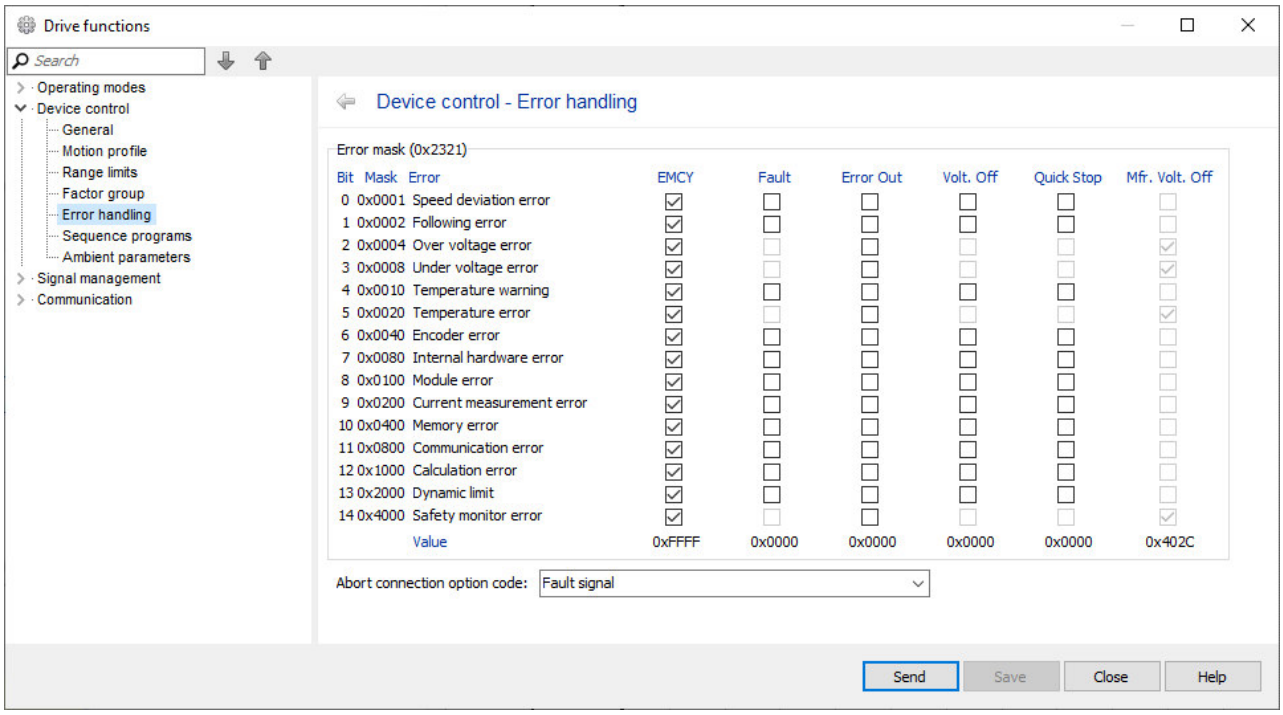


Figure 16 Configuration of the FAULT handling using the MotionManager

MC_Power_SML

The MC_Power_SML block is interacting with the control word and the status word to enable or disable the power-stage and control-loop. It requires to be connected to the correct instance of the axis structure – the very same which has been instanced by the SoftMotionLight axis. Additionally, there is a Boolean input which enables the FB to be evaluated and two Boolean inputs to start the drive and to enable the control -loop. As these FAULHABER low voltage drives don't need to be activated – not connected to the AC-grid by switching a contactor, we can use the same input variable for both.

There are some outputs for the MC_Power_SML block which would have to be used, if this was an automated machine, but that's not required in this simple example.

MC_Home_SML

Used to start and monitor the status of a drive-based homing sequence. The used homing method as well as any inputs used by the homing sequence must be configured in the drive beforehand using the MotionManager.

If the homing sequence doesn't finish but ends up with a homing error, one of the required inputs for the selected homing sequence might not have been configured in the drive. E.g. a lower limit switch needs to be configured in case of a homing 1 or homing 17 are to be used.

MC_MoveAbsolute_SML

The MC_MoveAbsolute_SML block is used to move a single axis in a Point to Point manner (PTP). To do so, it takes an absolute target position – in whatever scaling we configured for the SoftMotion axis – here µm. And it takes some profile parameters like max. speed, acceleration and deceleration. These values are

written into the drive where the drive-based profile generator is going to compute the intermediate position-, velocity- and torque demand values.

The block of course again needs to access the instance of the axis structure. Then there are the parameters for the motion itself and the binary “Execute” input which will start the motion at a rising edge of the input value.

Here two of the blocks where used to move to a Pos A on an external trigger and move back to Pos B directly after the first movement was finished. As these moves are absolute ones a proper adjustment of the position is usually mandatory. Which is why here the start of the movement to PosA is enabled only after a successful homing sequence.

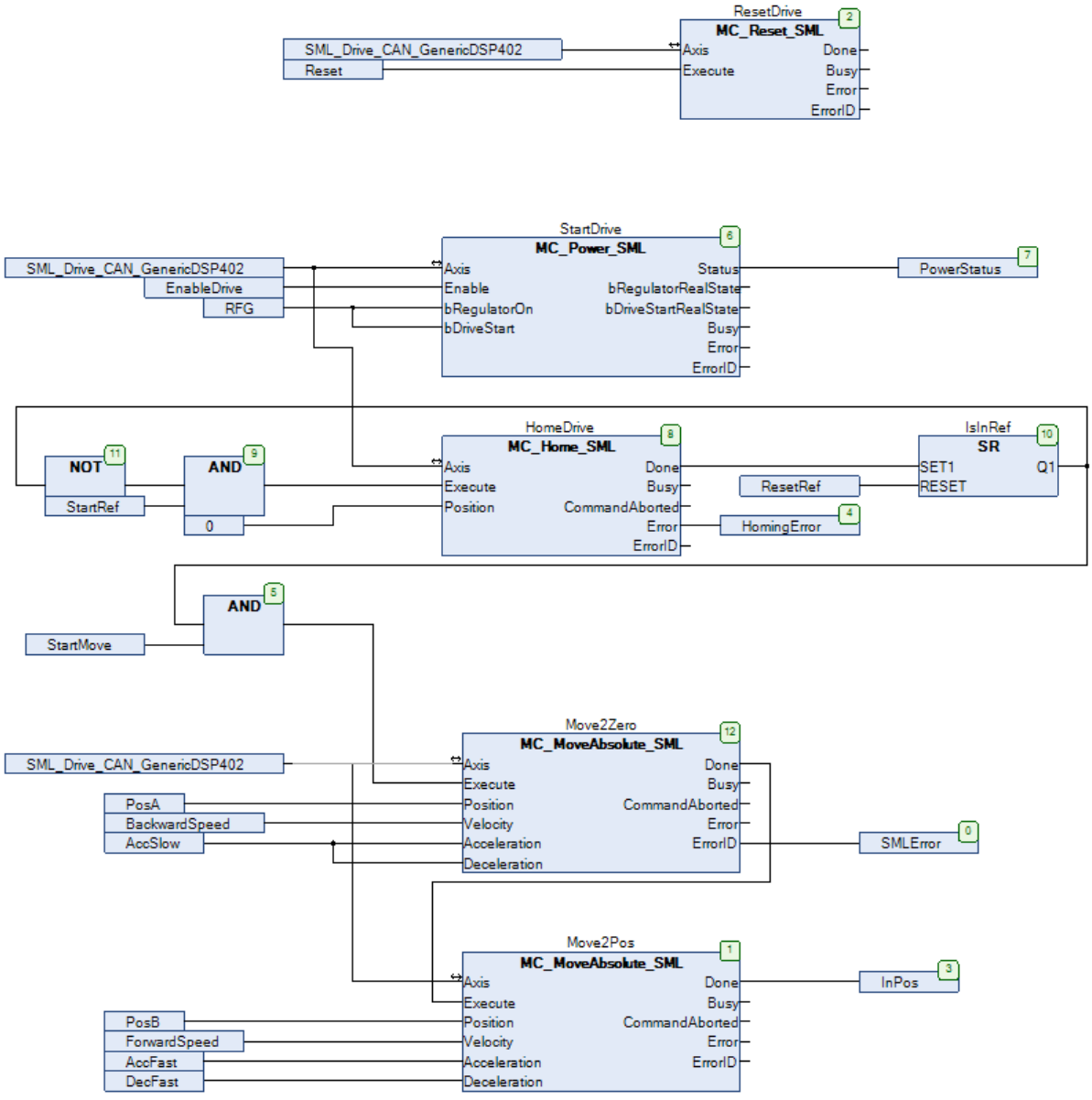


Figure 17 Demo application using the SML FBs out of the libray

Run the drive

Running the drive is controlled via a simple visualization (Figure 18).

Initially the power stage and the motor control has been enabled using the Switch On/Off button and the RFG button.

The required homing is started via “Start Homing” which of course requires a homing sequence to be selected and the inputs configured in the drive.

Afterwards the forward/backward moves can be started by the “Start Moving” button. A drive error can be acknowledged by means of the MC_Reset_SML FB and the “Reset Drive Error” button.

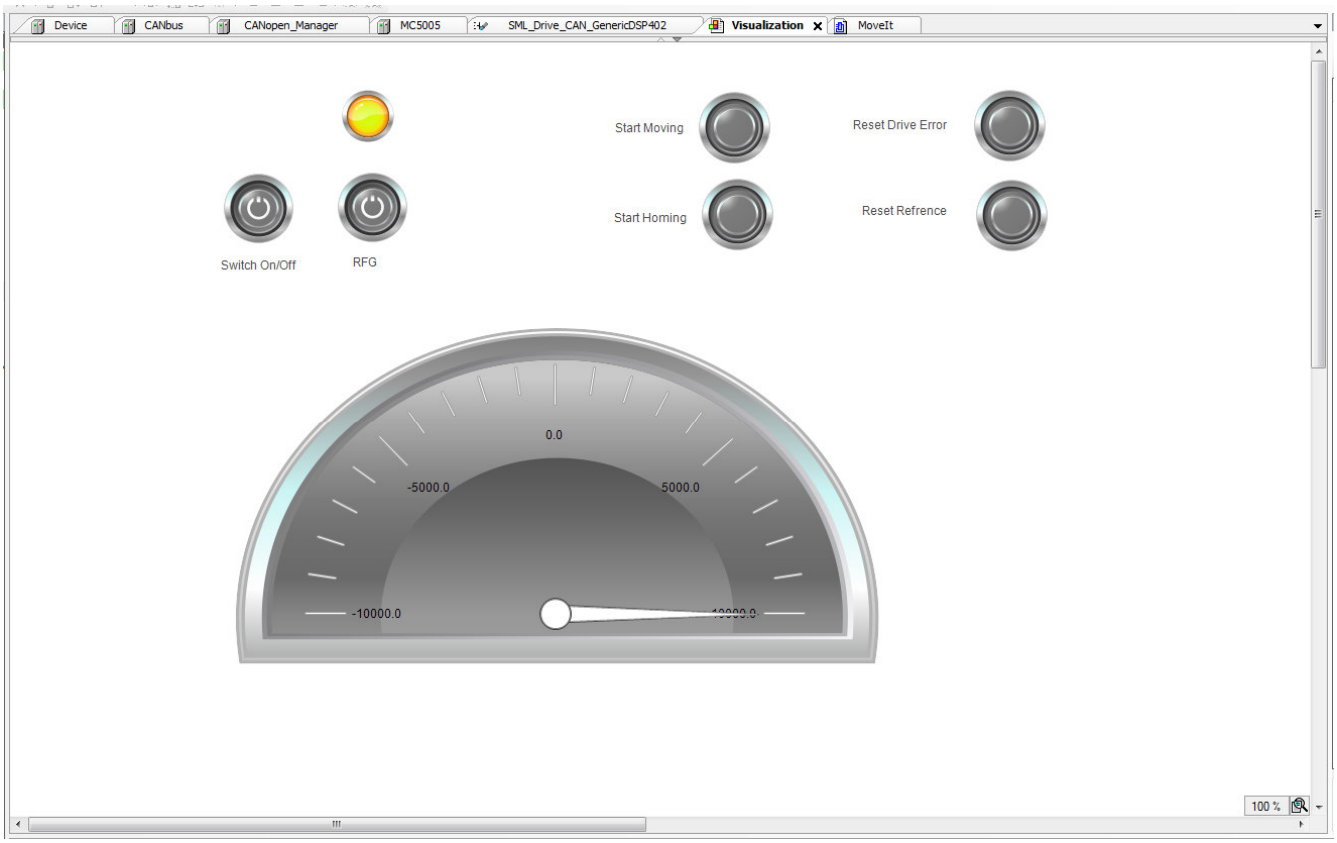


Figure 18 CODESYS visualization for the demo application

Additional Resources

FAULHABER Application Notes	
App-Note 174	Setup and configuration of a CANopen sub-system

FAULHABER manuals at www. faulhaber.com/manuals

FAULHABER demo systems at YouTube

Rechtliche Hinweise

Urheberrechte. Alle Rechte vorbehalten. Ohne vorherige ausdrückliche schriftliche Zustimmung der Dr. Fritz Faulhaber & Co. KG darf diese Application Note oder Teile dieser unabhängig von dem Zweck insbesondere nicht vervielfältigt, reproduziert, gespeichert (z.B. in einem Informationssystem) oder be- oder verarbeitet werden.

Gewerbliche Schutzrechte. Mit der Veröffentlichung, Übergabe/Übersendung oder sonstigen Zur-Verfügung-Stellung dieser Application Note werden weder ausdrücklich noch konkludent Rechte an gewerblichen Schutzrechten, übertragen noch Nutzungsrechte oder sonstige Rechte an diesen eingeräumt. Dies gilt insbesondere für gewerbliche Schutzrechte, die mittelbar oder unmittelbar den beschriebenen Anwendungen und/oder Funktionen dieser Application Note zugrunde liegen oder mit diesen in Zusammenhang stehen.

Kein Vertragsbestandteil; Unverbindlichkeit der Application Note. Die Application Note ist nicht Vertragsbestandteil von Verträgen, die die Dr. Fritz Faulhaber GmbH & Co. KG abschließt, und der Inhalt der Application Note stellt auch keine Beschaffenheitsangabe für Vertragsprodukte dar, soweit in den jeweiligen Verträgen nicht ausdrücklich etwas anderes vereinbart ist. Die Application Note beschreibt unverbindlich ein mögliches Anwendungsbeispiel. Die Dr. Fritz Faulhaber GmbH & Co. KG übernimmt insbesondere keine Gewährleistung oder Garantie dafür und steht auch insbesondere nicht dafür ein, dass die in der Application Note illustrierten Abläufe und Funktionen stets wie beschrieben aus- und durchgeführt werden können und dass die in der Application Note beschriebenen Abläufe und Funktionen in anderen Zusammenhängen und Umgebungen ohne zusätzliche Tests oder Modifikationen mit demselben Ergebnis umgesetzt werden können. Der Kunde und ein sonstiger Anwender müssen sich jeweils im Einzelfall vor Vertragsabschluss informieren, ob die Abläufe und Funktionen in ihrem Bereich anwendbar und umsetzbar sind.

Keine Haftung. Die Dr. Fritz Faulhaber GmbH & Co. KG weist darauf hin, dass aufgrund der Unverbindlichkeit der Application Note keine Haftung für Schäden übernommen wird, die auf die Application Note und deren Anwendung durch den Kunden oder sonstigen Anwender zurückgehen. Insbesondere können aus dieser Application Note und deren Anwendung keine Ansprüche aufgrund von Verletzungen von Schutzrechten Dritter, aufgrund von Mängeln oder sonstigen Problemen gegenüber der Dr. Fritz Faulhaber GmbH & Co. KG hergeleitet werden.

Änderungen der Application Note. Änderungen der Application Note sind vorbehalten. Die jeweils aktuelle Version dieser Application Note erhalten Sie von Dr. Fritz Faulhaber GmbH & Co. KG unter der Telefonnummer +49 7031 638 688 oder per Mail von mcsupport@faulhaber.de.

Legal notices

Copyrights. All rights reserved. This Application Note and parts thereof may in particular not be copied, reproduced, saved (e.g. in an information system), altered or processed in any way irrespective of the purpose without the express prior written consent of Dr. Fritz Faulhaber & Co. KG.

Industrial property rights. In publishing, handing over/dispatching or otherwise making available this Application Note Dr. Fritz Faulhaber & Co. KG does not expressly or implicitly grant any rights in industrial property rights nor does it transfer rights of use or other rights in such industrial property rights. This applies in particular to industrial property rights on which the applications and/or functions of this Application Note are directly or indirectly based or with which they are connected.

No part of contract; non-binding character of the Application Note. The Application Note is not a constituent part of contracts concluded by Dr. Fritz Faulhaber & Co. KG and the content of the Application Note does not constitute any contractual quality statement for products, unless expressly set out otherwise in the respective contracts. The Application Note is a non-binding description of a possible application. In particular Dr. Fritz Faulhaber & Co. KG does not warrant or guarantee and also makes no representation that the processes and functions illustrated in the Application Note can always be executed and implemented as described and that they can be used in other contexts and environments with the same result without additional tests or modifications. The customer and any user must inform themselves in each case before concluding a contract concerning a product whether the processes and functions are applicable and can be implemented in their scope and environment.

No liability. Owing to the non-binding character of the Application Note Dr. Fritz Faulhaber & Co. KG will not accept any liability for losses arising from its application by customers and other users. In particular, this Application Note and its use cannot give rise to any claims based on infringements of industrial property rights of third parties, due to defects or other problems as against Dr. Fritz Faulhaber GmbH & Co. KG.

Amendments to the Application Note. Dr. Fritz Faulhaber & Co. KG reserves the right to amend Application Notes. The current version of this Application Note may be obtained from Dr. Fritz Faulhaber & Co. KG by calling +49 7031 638 688 or sending an e-mail to mcsupport@faulhaber.de.