

**Instruction Manual**  
**JS\_MC\_LIB with XENAX® Xvi,**  
**EtherCAT® Busmodule and Codesys.**

Version 1.1.6

Edition 19. July 2024



**EtherCAT®** 

XENAX® Ethernet servo controller with  
EtherCAT® Busmodul

Functional Safety, TÜV certified.  
Force processes with „Force Limitation“,  
„Force Monitoring“ and „Force Control“.

## **Introduction**

This manual describes the **Jenny Science Motion Control Library (JS\_MC\_Lib)** for Codesys. This library is designed after the PLCopen standard but also integrates Jenny Science specific features.

## Contents

<b>1 Known issues</b>	<b>5</b>
<b>2 Development Environment</b>	<b>6</b>
2.1 Codesys GMBH	6
2.1.1 Programmable Logic Controller	6
2.1.2 Codesys	6
2.2 Jenny Science	6
2.2.1 WebMotion	6
2.2.2 XENAX® servo controller	7
2.2.3 LINAX® Linear motors	7
2.2.4 ELAX® Linear motor slides	7
2.2.5 ROTAX® Rotary motor axes	7
2.3 Required Resources	8
2.3.1 ESI EtherCAT Slave Information	8
2.3.2 JS_MC_Lib	8
2.4 Software Requirements	8
<b>3 PLCopen Library (JsMcLib)</b>	<b>9</b>
3.1 Drive Modes: point to point or interpolated	9
3.2 State Diagram	10
3.2.1 Profile Position Mode	11
3.2.2 Cyclic Synchronous Position Mode	12
3.3 Function blocks required for operation	13
3.3.1 JS_MC_Init	13
3.3.2 JS_MC_CyclicIn	14
3.3.3 JS_MC_CyclicOut	14
3.3.4 JS_MC_Power	14
3.3.5 JS_MC_Reference	15
3.3.1 JS_MC_Reset	16
3.3.1 JS_MC_Stop	16
3.4 Additional function blocks for Forceteq®	17
3.4.1 JS_MC_ForceCalibration	17
3.4.1 JS_MC_WriteLimit_I_Force	17
3.4.1 JS_MC_Read_I_Force	18
3.4.2 JS_MC_WriteLimit_Force	18
3.4.3 JS_MC_Read_Force	18
3.5 Additional function blocks for cyclic synchronous position mode	19
3.6 Additional function blocks for Profile Position	20
3.6.1 JS_MC_MoveAbsolute	20
3.6.2 JS_MC_MoveRelative	20

3.6.3 JS_MC_JogVelocity	21
3.6.4 JS_MC_Halt	21
3.1 Function blocks for error handling	22
3.1.1 JS_MC_ReadAxisError	22
3.1.2 JS_MC_ReadLibraryError	23
3.2 Optional function blocks	24
3.2.1 JS_MC_ReadStatus	24
3.2.2 JS_MC_ReadPSR	24
3.2.3 JS_MC_WriteDigitalOutput	25
3.2.4 JS_MC_ReadActualPosition	25
3.2.1 JS_MC_ReadDigitalInput	25
3.2.2 JS_MC_WriteParameter	26
3.2.3 JS_MC_ReadParameter	26
3.3 Minimum and Maximum Values of Function Blocks	27
3.4 Error numbers	27
3.5 Error sources	30
3.6 Error type	30
<b>4 Example Projects in Codesys</b>	<b>31</b>
4.1 Open Project	32
4.2 XENAX® ESI XML Installation	32
4.3 Library Installation	33
4.4 Download Programm	33
4.5 Launch Demo Program	34
4.5.1 Simple Demo	34
4.5.2 Force Limit	34
4.5.3 Force Sectors	34
<b>5 New Project profile position in Codesys</b>	<b>36</b>
5.1 Create Project	36
5.2 Add EtherCAT Master	37
5.3 Add XENAX®	38
5.1 Add Jenny Science Library	38
5.2 Copy Code	39
5.3 PDO Mapping	39
5.4 PDO Linking	39
5.5 Launch Project	39
<b>6 New Project Softmotion CSP in Codesys</b>	<b>40</b>
6.1 Create Project	40
6.2 Add EtherCAT Master	41
6.3 Add XENAX® as EtherCAT Slave	42

6.4 Configure XENAX®	42
6.5 Add Softmotion Axis	44
6.6 Add Program Code	46
6.7 PDO Linking	47
6.8 Download the Project	47
<b>7 Version history</b>	<b>48</b>

## 1 Known issues

We do not recommend using Softmotion (CSP mode) with Forceteq functions. Using Profile Position Mode is recommended. There is an issue with drive commands with limited current when the axis drives into an obstacle and should drive back on touch. During change of direction, the reset following error does not work correctly. Codesys changes the set position (target position) when it shouldn't. This leads to a drop of the motor current when it should stay constant.

We are investigating this issue with Codesys.

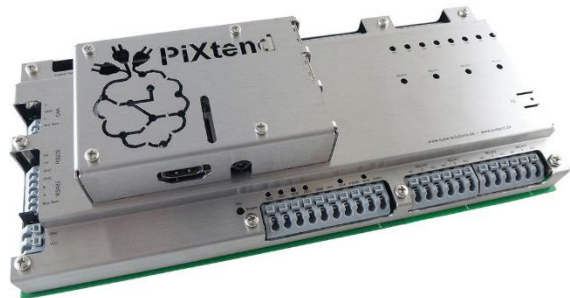
## 2 Development Environment

### 2.1 Codesys GMBH

#### 2.1.1 Programmable Logic Controller

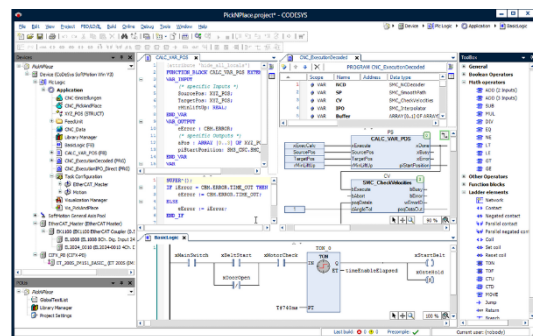
Codesys compatible PLC with EtherCAT interface is required.

Hint: If the PLC is running on a Windows machine, it is recommended to deactivate Vtx and secure boot in the Bios option. Otherwise, the PLC may not start.



#### 2.1.2 Codesys

Codesys is the software development platform used to program PLCs.



## 2.2 Jenny Science

### 2.2.1 WebMotion

The proprietary graphical user interface for servo controllers is stored in the embedded web server of the XENAX® servo controller

WebMotion® is launched with a web browser by entering the corresponding TCP/IP address of XENAX®.

LINAX® linear motor axes, ELAX® linear motor slides and ROTAX® rotary axes are automatically recognized. The corresponding controller parameters are saved and loaded automatically. With the Quick Start button, the linear motors can easily and immediately be operated. No other user manual is needed.



## 2.2.2 XENAX® servo controller

XENAX® servo controller for Jenny Science Axis with integrated EtherCAT bus module. The bus module is optional but it is required for this application. One XENAX® can control one axis. The XENAX® servo controller recognises all Jenny Science motors and configures the parameters correctly.



## 2.2.3 LINAX® Linear motors

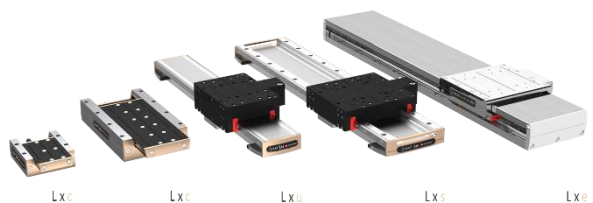
The LINAX® linear motor axes are highly modular and can be flexibly combined amongst each other. Four different series are available.

Lxc = compact

Lxu = universal

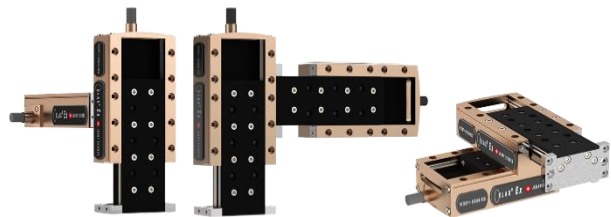
Lxs = shuttle

Lxe = exclusive



## 2.2.4 ELAX® Linear motor slides

Specifically designed for handling and Pick and Place tasks with strokes from 30mm up to 150mm. The configuration is extremely modular and there is only one cable to the XENAX® servo controller.



## 2.2.5 ROTAX® Rotary motor axes

Specifically designed for fast and precise assembly and handling tasks. It can be equipped with standard gripping tools which enables a 360° rotation and has a hollow shaft feedthrough for vacuum or compressed air.

Rxvp = vacuum pressure

Rxhq = high torque



## 2.3 Required Resources

The following resources are needed for the successful operation of the XENAX® servo controller with an EtherCAT bus module.

### 2.3.1 ESI EtherCAT Slave Information

The ESI XML file is available on our website [www.jennyscience.com](http://www.jennyscience.com) under XENAX® → Firmware Bus Module.

### 2.3.2 JS\_MC\_Lib

The Jenny Science Motion Control library is included in the zip file of this manual or it can be downloaded from our website [www.jennyscience.com](http://www.jennyscience.com) under XENAX® → PLCopen Library.

## 2.4 Software Requirements

Software	Version
Codesys	V3.5 SP17 (older versions should work, but are not tested)
XENAX Firmware	V5.20 or later
EtherCAT Bus-Module	V2.70 or later



### 3 PLCopen Library (JsMcLib)

Jenny Science provides a PLCopen library for Codesys. The PLCopen standard is easy to understand and includes basic movement functions as well as Jenny Science specific features.

#### 3.1 Drive Modes: point to point or interpolated

The Jenny Science Motion Control Library supports two fundamental different drive modes.

##### 1. Point to point = Profile Position Mode:

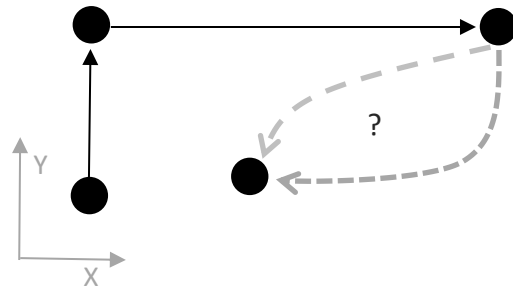
The parameters distance, speed, acceleration and s-curve are fixed before a drive. The trajectory (driving curve) is calculated on the Xenax®. This driving mode is simpler to implement, but gives less control over the driving curve to the PLC. It is not possible drive a straight line with a XY-Axis since both Axis can be started at the same time but will reach their target at different times. It is also not possible to drive along a round curve because only the target position can be specified and not the path to the target location.

This mode fits a small PLC with low performance. There is **no** need for a **virtual nc-axis**.

##### 2. Interpolated = Cyclic Synchronous Position Mode:

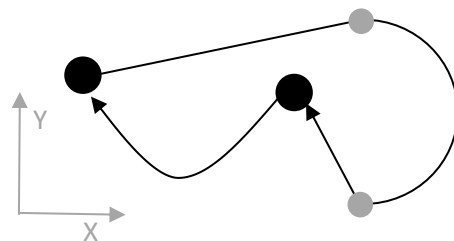
In the cyclic synchronous position mode, the target position is passed to the XENAX® servo controller at cyclic time intervals (for example every millisecond). The trajectory (driving curve) is calculated on the Rexroth PLC. For this reason, a virtual Axis for each Axis is needed. This enables full control over the driving curve. Thanks to the **virtual nc-axis**, round curves or other complex driving paths are now possible.

XY-Axis Profile Position



Limited control over driving path between two target positions with different X and Y coordinates. Furthermore, the speed and target position can not be changed during a drive. An Axis has to stop at every target position.

XY-Axis Cyclic Synchronous Position



Full control over Axis movement. Two grey circles show a change in direction and speed without a stop.

### 3.2 State Diagram

The following diagram shows the state and the behaviour of the axis when multiple motion control function blocks are “simultaneously” active.

Each motion command is a transition that changes the state of the axis and, as a consequence, influences the method of calculation of the current movement.

All function blocks which do not appear in the state diagram, do not affect the state of the axis.

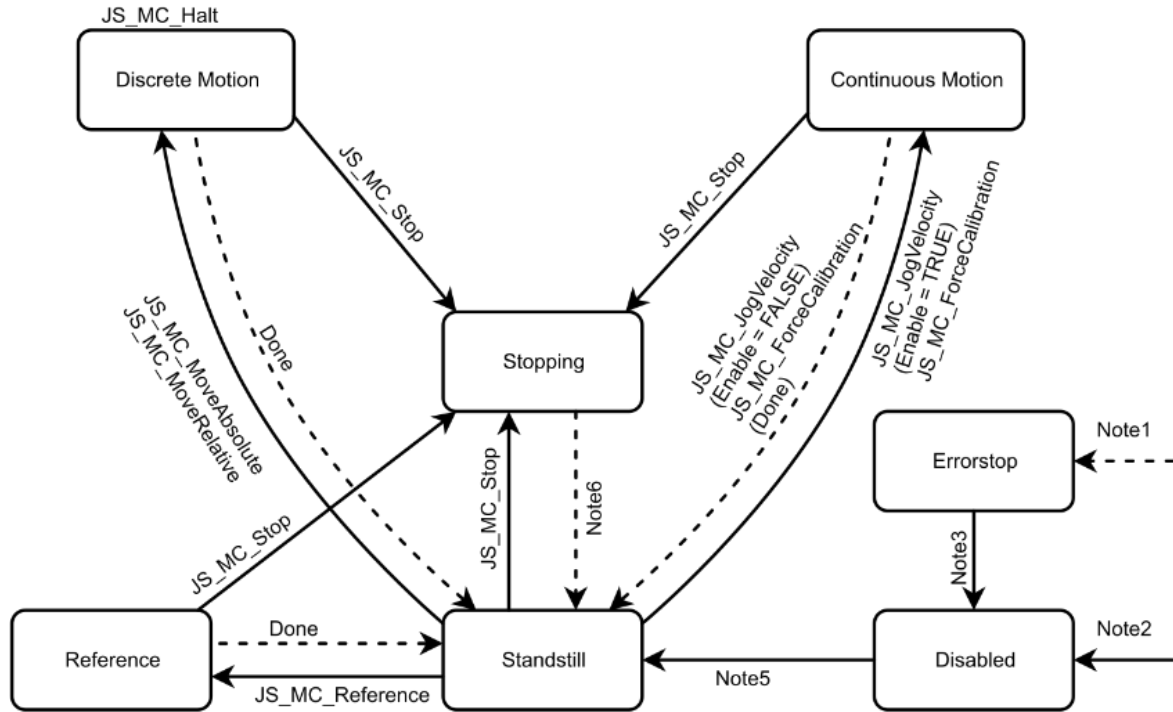
The current state of the axis can be determined with the function block “**JS\_MC\_ReadStatus**”. If a function block is called where it is not allowed, the function block reports an error.

The notes describe the necessary conditions that must be met for a change in an axis state.

**Important:**

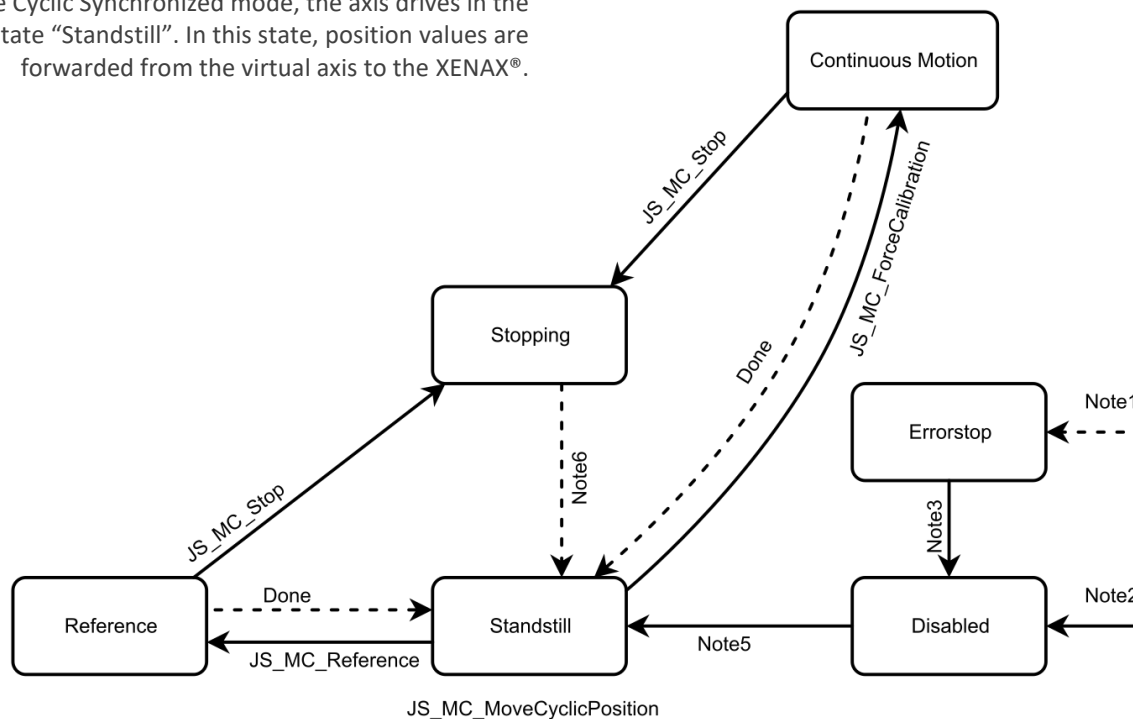
In the states “**Stopping**”, “**ErrorStop**”, “**Disabled**” and “**Reference**” no motion blocks can be called. In standstill condition, an axis must always be referenced before starting a movement.

### 3.2.1 Profile Position Mode



### 3.2.2 Cyclic Synchronous Position Mode

In the Cyclic Synchronized mode, the axis drives in the state "Standstill". In this state, position values are forwarded from the virtual axis to the XENAX®.



Note 1:

From any state. An error in the axis occurred.

Note 2:

From any state. JS\_MC\_Power.Enable = FALSE and there is no error in the axis.

Note 3:

```
JS_MC_Reset AND JS_MC_Power.Status = FALSE.
```

Note 5:

```
JS_MC_Power.Enable = TRUE AND
JS_MC_Power.Status = TRUE
```

Note 6:

```
JS_MC_Stop.Done = TRUE AND JS_MC_Stop.Execute = FALSE
```

### 3.3 Function blocks required for operation

The functionality of the JsMcLib is implemented in various small function blocks. In this subchapter, all those function blocks are described. Demo programs in the subsequent chapters will show the function blocks in action.

If Codesys has a similar function block, the JsMcLib function block is implemented based on the original block. Note that Units are always in increments. Velocity is in increments per seconds while acceleration and deceleration are in increments per second<sup>2</sup>.

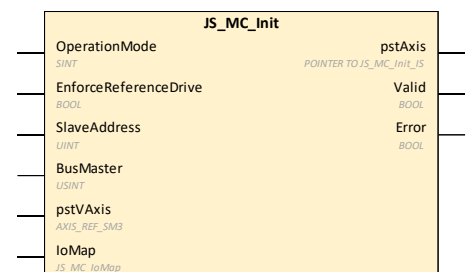
Function Block Input Name	Unit
Position	Increments
Velocity	Increments/s
Acceleration	Increments/s <sup>2</sup>
Deceleration	Increments/s <sup>2</sup>

#### 3.3.1 JS\_MC\_Init

This function block must be called once at start up to initialize library variables. The block also provides a reference to the axis which is needed in all other JS\_MC\_Lib function blocks. Calling this function block a second time will only re-evaluate the Axis output reference.

Inputs	
OperationMode	Choose gcJS_MC.jsmc_MODE_CYCLIC_SYNC if you use a NC-Axis, otherwise use gcJS_MC.jsmc_Mode_PROFILE_POSITION
EnforceReferenceDrive	Set this to TRUE if motor must execute a reference drive even though they are already referenced. Recommended = FALSE
SlaveAddress	EtherCAT address or the current AutoInc address at the bus (default Address type is ECAT_AUTOINC_ADDRESS)
pstVAxis	Pointer to Virtual Softmotion axis, only used for cyclic synchronous position mode
IoMap	IO mapping structur, map axis PDOs to this input structure

Outputs	
pstAxis	The axis reference handle
Valid	A valid reference handle of the axis is available
Error	Error occurred within function block

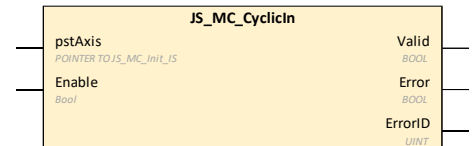


### 3.3.2 JS\_MC\_CyclicIn

This function block has to be called at the start of the periodically called program.  
Checks if communication with axis is valid.  
Reads cyclic data from the fieldbus.

Inputs	
pstAxis	The axis reference handle
Enable	As long as "Enable" is TRUE, cyclic data are received from the bus. (Must always be TRUE during operation)

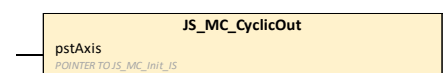
Outputs	
Valid	Cyclic realtime bus communication is valid (it is not allowed to enable or execute other function blocks unless valid is TRUE)
Error	Error occurred within function block
ErrorID	Error number



### 3.3.3 JS\_MC\_CyclicOut

Important: All other JSC\_MC\_Lib blocks must be called between CyclicIn and CyclicOut.  
Writes cyclic data to the fieldbus.

Inputs	
pstAxis	The axis reference handle

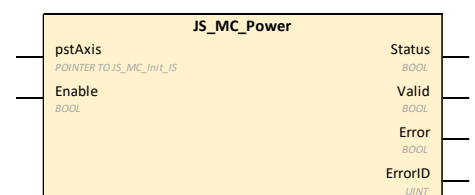


### 3.3.4 JS\_MC\_Power

This function block switches the power stage on and off.

Inputs	
pstAxis	The axis reference handle
Enable	As long as "Enable" is TRUE, the drive power Stage is enabled

Outputs	
Status	Effective status of the power Stage
Valid	TRUE when the axis is ready for operation
Error	Error occurred within function block
ErrorID	Error number

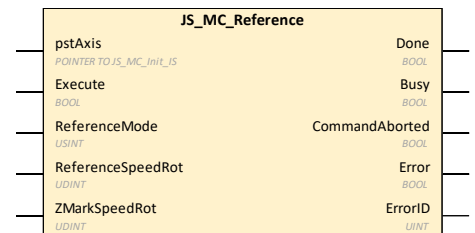


### 3.3.5 JS\_MC\_Reference

Performs a reference drive. The goal of the reference drive is to find the absolute position of the axis. The axis either drives to a mechanical stopper or to a Z-Mark indicator on the scale. The ReferenceMode influences the behaviour during a reference drive.

Inputs	
pstAxis	The axis reference handle
Execute	Start reference at rising edge
ReferenceMode	Select behaviour during reference drive
ReferenceSpeedRot	Reference speed towards a reference switch [increment/s] (only for rotative drives)
ZMarkSpeedRot	Reference speed towards a Z-Mark on the scale [increment/s] (only for rotative drives)

Outputs	
Done	Reference procedure has finished successfully
Busy	The function block is not finished
CommandAborted	Function block is aborted by another command
Error	Error occurred within function block
ErrorID	Error number



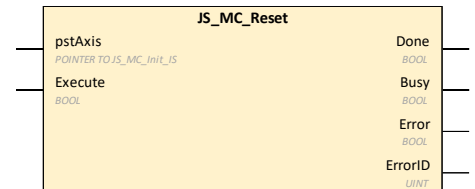
### 3.3.1 JS\_MC\_Reset

This function block makes a transition in the State Diagram from Errorstop to Standstill or Disabled by resetting the axis error.

Inputs	
pstAxis	The axis reference handle
Execute	Reset the axis at the rising edge

Outputs	
Done	Standstill or Disabled state is reached
Busy	The function block is not finished
Error	Error occurred within function block
ErrorID	Error number



### 3.3.1 JS\_MC\_Stop

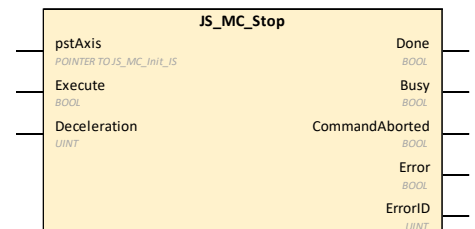
This function block commands a controlled motion stop and transfers the axis to the state Stopping. The axis stays in the Stopping state until execute is set back to FALSE. All move commands are blocked as long as the axis stays in the stopping state.

The axis does not have to be in motion to call MC\_Stop. This means MC\_Stop can be used to ensure an axis stays at the same position.

Inputs	
pstAxis	The axis reference handle
Execute	Reset the action at rising edge
Deceleration	Value of the deceleration [inc/s <sup>2</sup> ]

Outputs	
Done	Zero velocity is reached
Busy	The function block is not finished
CommandAborted	Command is aborted by switching off power (only possibility to abort)
Error	Error occurred within function block
ErrorID	Error number





### 3.4 Additional function blocks for Forceteq®

#### 3.4.1 JS\_MC\_ForceCalibration

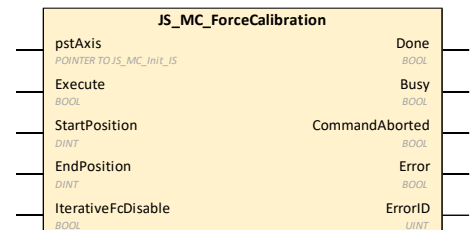
Starts a Force Calibration. The axis moves from start- to end position and measures cogging force and friction. Those two forces are then compensated in future drives.

An active Force Calibration can only be stopped by calling JS\_MC\_STOP.

If the motor oscillates during the Force Calibration, set IterativeFcDisable. This will clear the old calibration data before a new calibration is started.

Inputs	
pstAxis	The axis reference handle
Execute	Start move at rising edge
StartPosition	Start position for the force calibration [Inc]
EndPosition	End position for the force calibration [Inc]
IterativeFcDisable	0 = Takes values from previous FC to improve calibration 1= Ignores calibration values from last FC

Outputs	
Done	Force Calibration finished successfully
Busy	Function block is not finished
CommandAborted	Function block is aborted by another command
Error	Error occurred within function block
ErrorID	Error number

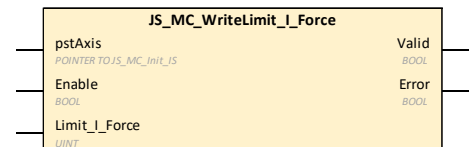


#### 3.4.1 JS\_MC\_WriteLimit\_I\_Force

Sets the I\_Force limitation in [10mA] with Forceteq Basic. Note that the PDO Limit\_I\_Force must be mapped and linked.

Inputs	
pstAxis	The axis reference handle
Enable	Writes Limit I_Force to XENAX when enabled
Limit_I_Force	I_Force limitation in [x10 mA], 20 = 200mA , 0 = no limitation.

Outputs	
Valid	Write was successful
Error	Error occurred within function block

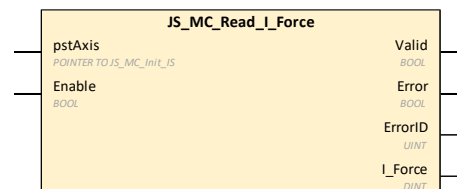


### 3.4.1 JS\_MC\_Read\_I\_Force

Reads the Force-proportional I\_Force in mA with Forceteq® basic.

Inputs	
pstAxis	The axis reference handle
Enable	As long as "Enable" is TRUE, the actual I_Force is read out continuously

Outputs	
Valid	A valid set of outputs is available at the function block
Error	Error occurred within function block
ErrorID	Error number
I_Force	Actual I_Force [mA]

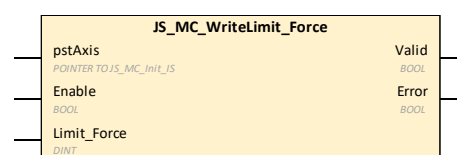


### 3.4.2 JS\_MC\_WriteLimit\_Force

Sets the force limitation in mN based on the value measured by the Signateq® force sensor (only for Forceteq Pro). Note that the PDO Limit\_Force must be mapped and linked.

Inputs	
pstAxis	The axis reference handle
Enable	Writes LimitForce to XENAX when enabled
Limit_Force	Force limit [mN]

Outputs	
Valid	Last write was successful
Error	Error occurred within function block

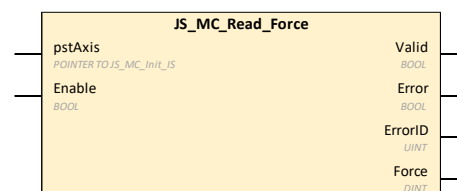


### 3.4.3 JS\_MC\_Read\_Force

Reads the force in mN measured by the Signateq® force sensor (only with Forceteq Pro).

Inputs	
pstAxis	The axis reference handle
Enable	As long as "Enable" is TRUE, the actual force is read out continuously

Outputs	
Valid	A valid set of outputs is available at the function block
Error	Error occurred within function block
ErrorID	Error number
Force	Actual force measured by Signateq® [mN]



### 3.5 Additional function blocks for cyclic synchronous position mode

Use the MC\_MoveAbsolute function block from Codesys to move the axis.

JS\_MC\_STOP calls MC\_STOP internally if the axis is driving in cyclic synchronous position mode (csp).

During a force calibration or a reference drive, the axis is not driving in csp mode. JS\_MC\_STOP will still correctly abort a non csp mode drive. This means that only JS\_MC\_STOP will abort a force calibration or a reference drive and MC\_STOP will not.

### 3.6 Additional function blocks for Profile Position

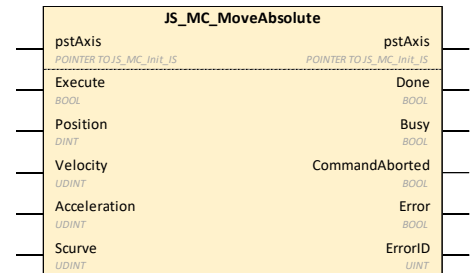
#### 3.6.1 JS\_MC\_MoveAbsolute

This function block drives to an absolute position.  
Only for profile position mode. In cyclic synchronous position mode, use MC\_MoveAbsolute instead.

Inputs	
Execute	Start move at rising edge
Position	Target position for the motion [inc]
Velocity	Value of maximum velocity [inc/s] (not necessarily reached)
Acceleration	Value of maximum acceleration [inc/s <sup>2</sup> ] (not necessarily reached)
Scurve	Value of S-curve parameter during the acceleration [%]

Outputs	
Done	Commanded position reached
Busy	The function block is not finished
CommandAborted	Function block is aborted by another command
Error	Error occurred within function block
ErrorID	Error number

IN/OUT	
pstAxis	The axis reference handle



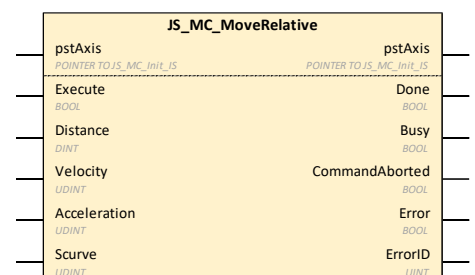
#### 3.6.2 JS\_MC\_MoveRelative

This function block drives to a relative position.  
Only for profile position mode. In cyclic synchronous position mode, use MC\_MoveRelative instead.

Inputs	
Execute	Start move at rising edge
Distance	Target distance for the motion [inc]
Velocity	Value of maximum velocity [inc/s] (not necessarily reached)
Acceleration	Value of maximum acceleration [inc/s <sup>2</sup> ] (not necessarily reached)
Scurve	Value of S-curve parameter during the acceleration [%]

Outputs	
Done	Commanded position reached
Busy	The function block is not finished
CommandAborted	Function block is aborted by another command
Error	Error occurred within function block
ErrorID	Error number

IN/OUT	
pstAxis	The axis reference handle



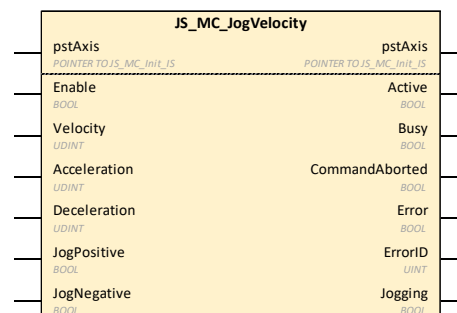
### 3.6.3 JS\_MC\_JogVelocity

This function block drives with a constant speed in positive or negative direction. Only for profile position mode. In cyclic synchronous position mode, use MC\_JogVelocity instead.

Inputs	
Enable	As long as "Enable" is TRUE, the axis in in Continuous Motion mode (Jog drive possible)
Velocity	Value of maximum velocity [inc/s] Note: This value can also be changed while a movement is taking place
Acceleration	Value of maximum acceleration [inc/s <sup>2</sup> ] Note: This value can also be changed while a movement is taking place (new value is used at the next velocity change)
Deceleration	Value of maximum deceleration [inc/s <sup>2</sup> ] Note: This value can also be changed while a movement is taking place (new value is used at the next velocity change)
JogPositive	Executes a movement in the positive direction
JogNegative	Executes a movement in the negative direction

Outputs	
Active	The function block is active, possible to execute movements
Busy	The function block is not finished
CommandAborted	Function block is aborted by another command
Error	Error occurred within function block
ErrorID	Error number
Jogging	Movement being carried out

IN/OUT	
pstAxis	The axis reference handle



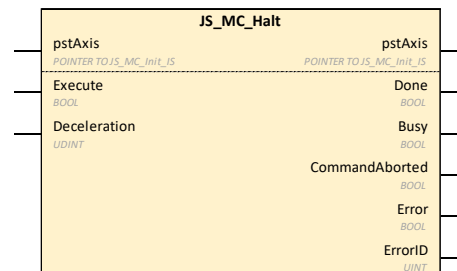
### 3.6.4 JS\_MC\_Halt

This function block stops an ongoing JS\_MC\_MoveAbsolute or a JS\_MC\_MoveRelative command and switches to the state standstill.

Inputs	
Execute	Start the action at rising edge
Deceleration	Value of the deceleration [inc/s <sup>2</sup> ]

Outputs	
Done	Zero velocity is reached
Busy	Function block is not finished
CommandAborted	Function block is aborted by another command
Error	Error occurred within function block
ErrorID	Error number

IN/OUT	
pstAxis	The axis reference handle



### 3.1 Function blocks for error handling

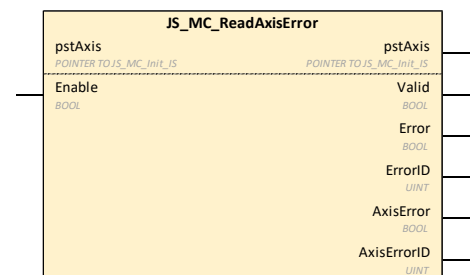
#### 3.1.1 JS\_MC\_ReadAxisError

Describes general axis errors that are not related to function blocks. Use JS\_MC\_RESET to clear the axis error.

Inputs	
Enable	While TRUE, the output value provides the parameter value continuously for reading out.

Outputs	
Valid	Valid outputs are available
Error	Error occurred within function block
ErrorID	Error number
AxisError	Axis error has occurred.
AxisErrorID	Axis error number, see XENAX manual

IN/OUT	
pstAxis	The axis reference handle



### 3.1.2 JS\_MC\_ReadLibraryError

This block can be used to handle all errors at a central point in the code. Note that this function block is completely optional.

Collect all errors from axis and JS\_MC\_Lib function blocks. All errors are collected in a queue. The first error in the queue is displayed in the output ErrorRecord.

Each error must be acknowledged to display the next error until the queue is empty.

The error should be handled first before it is acknowledged. This means to reset the enable or execute input of the

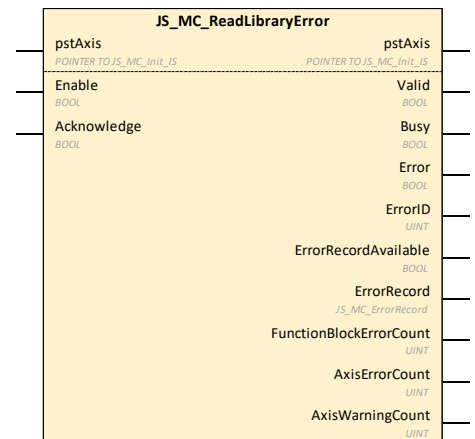
Function block which caused the error. Or to call JS\_MC\_RESET to clear the error caused by the axis.

An unhandled error which is acknowledged will be recollected and again saved in the queue.

Inputs	
Enable	As long as "Enable" is TRUE, the function block can be used to read out axis and function block errors
Acknowledge	Acknowledges the error record currently displayed

Outputs	
Valid	A valid set of outputs is available at the function block. This output is set to FALSE while an error is being acknowledged or error text is being read
Busy	New output data is to be expected. This output is set to TRUE while an error is being acknowledged or error text is being read
Error	Error occurred within this function block
ErrorID	Error number of this function block
ErrorRecordAvailable	Set if a new error record is displayed in the "ErrorRecord" output. FALSE if errpr queue is empty
ErrorRecord	Displays the first error in the queue including the error number, error type and error source.
FunctionBlockErrorCount	Number of pending function block errors to display
AxisErrorCount	Number of pending axis errors to display
AxisWarningCount	Number of pending axis warnings to display

IN/OUT	
pstAxis	The axis reference handle



## 3.2 Optional function blocks

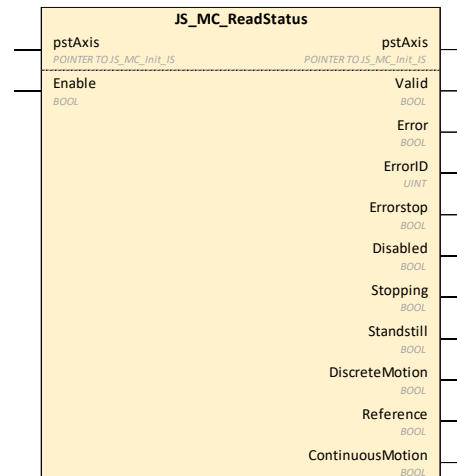
### 3.2.1 JS\_MC\_ReadStatus

Reads the current state of the PLCopen DS402 state machine.

Inputs	
Enable	As long as "Enable" is TRUE, the axis status is read out continuously

Outputs	
Valid	A valid set of outputs is available at the function block
Error	Error occurred within function block
ErrorID	Error number
Errorstop	An error has occurred. Use JS_MC_Reset to acknowledge errors
Disabled	JS_MC_Power has not powered the axis, or an error has been acknowledged by JS_MC_Reset and the axis has been turned off
Stopping	JS_MC_Stop is active
Standstill	Motion is not active on the drive
DiscreteMotion	Axis is in motion due to one of the following function blocks: JS_MC_MoveAbsolute, JS_MC_Relative.
Reference	JS_MC_Reference has started referencing the axis
ContinuousMotion	Axis is in motion due to the following function block: JS_MC_JogVelocity, JS_MC_ForceCalibration

IN/OUT	
pstAxis	The axis reference handle



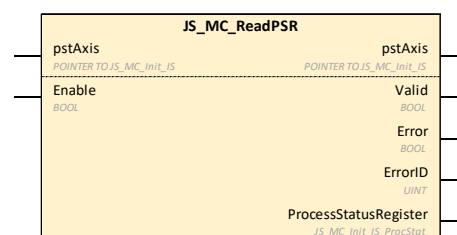
### 3.2.2 JS\_MC\_ReadPSR

Reads the Process Status Register (PSR). This registers contains various information about the XENAX® servo controller.

Inputs	
Enable	As long as "Enable" is TRUE, the actual Process Status Register is read out continuously

Outputs	
Valid	A valid set of outputs is available at the function block
Error	Error occurred within function block
ErrorID	Error number
ProcessStatusRegister	Process Status Register of the XENAX controller (see data type JS_MC_Init_IS_ProcStat)

IN/OUT	
pstAxis	The axis reference handle





## 3.2.3 JS\_MC\_WriteDigitalOutput

This function block sets the digital output pins on the servocontroller. The output is written once when "Execute" is set.

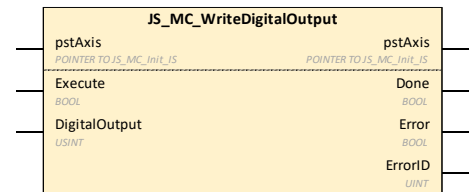
Inputs	
Execute	Writes the Digital value at the rising edge
DigitalOutput	The value of digital outputs (bit-coded)

Outputs	
Done	Digital outputs are written
Error	Error occurred within function block
ErrorID	Error number

IN/OUT	
pstAxis	The axis reference handle



## 3.2.4 JS\_MC\_ReadActualPosition

Reads the position of the axis in increments. Note that actual position PDO must be mapped.

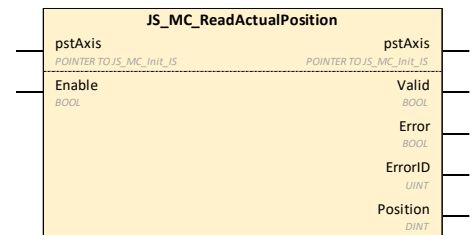
Inputs	
Enable	As long as "Enable" is TRUE, the actual position is read continuously

Outputs	
Valid	A valid set of outputs is available at the function block
Error	Error occurred within function block
ErrorID	Error number
Position	Actual position of the axis [Inc]

IN/OUT	
pstAxis	The axis reference handle



## 3.2.1 JS\_MC\_ReadDigitalInput

Reads digital inputs which are located in the XENAX socket. Note that "Digital Inputs" PDO must be mapped.

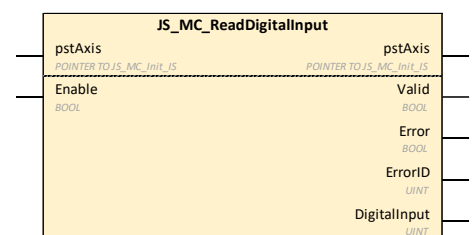
Inputs	
Enable	As long as "Enable" is TRUE, the digital inputs are read continuously

Outputs	
Valid	A valid set of outputs is available at the function block
Error	Error occurred within function block
ErrorID	Error number
DigitalInput	The value of digital inputs (bit-coded)

IN/OUT	
pstAxis	The axis reference handle



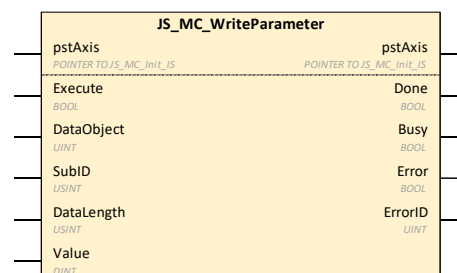
### 3.2.2 JS\_MC\_WriteParameter

This block is used to write CANopen parameter from the axis. All available CANopen parameters are described in the "CANOPEN\_ETHERNET\_MANUAL.pdf" (<https://www.jennyscience.ch/en/products/download>).

Inputs	
Execute	Writes the value at the rising edge
DataObject	Desired data object to be written (according CANopen communication profile)
SubID	SubID of the desired data object to be written
DataLength	Data length of the desired data object to be written in bytes
Value	Value to be written to the desired data object

Outputs	
Done	Value is written to the object
Busy	The function block is not finished
Error	Error occurred within function block
ErrorID	Error number

IN/OUT	
pstAxis	The axis reference handle



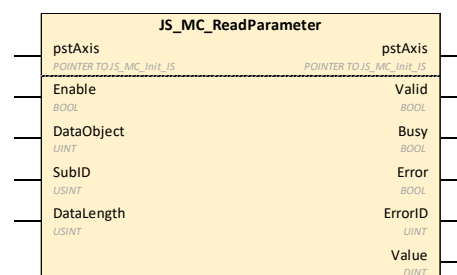
### 3.2.3 JS\_MC\_ReadParameter

This block is used to read out CANopen parameter from the axis. All available CANopen parameters are described in the "CANOPEN\_ETHERNET\_MANUAL.pdf" (<https://www.jennyscience.ch/en/products/download>).

Inputs	
Enable	As long as "Enable" is TRUE, the desired data object is read out continuously
DataObject	Desired data object to be written (according CANopen communication profile)
SubID	SubID of the desired data object to be written
DataLength	Data length of the desired data object to be written in bytes

Outputs	
Valid	A valid set of outputs is available at the function block
Busy	The function block is not finished
Error	Error occurred within function block
ErrorID	Error number
Value	Value to be written to the desired data object

IN/OUT	
pstAxis	The axis reference handle



### 3.3 Minimum and Maximum Values of Function Blocks

Following minimum and maximum values of the function blocks should be adhered to.

name	datatype	min	max
<b>Velocity linear</b>	UDINT	10 inc/s	9000000 inc/s
<b>Velocity rotative</b>	UDINT	10 inc/s	100000000 inc/s
<b>Deceleration</b>	UDINT	2000 inc/s <sup>2</sup>	1000000000 inc/s <sup>2</sup>
<b>Acceleration</b>	UDINT	2000 inc/s <sup>2</sup>	1000000000 inc/s <sup>2</sup>
<b>S-curve</b>	UDINT	1 %	100 %

### 3.4 Error numbers

The following ErrorIDs can be generated by the JsMclib function blocks. Lower numbers than 5000 are Axis Error generated by the XENAX® servo controller. Please look up those errors in the XENAX® Manual.

Value	Name	Description	Correction
0	ERR_OK	FUB executed correctly with no errors	None.
50000	jsmcERR_NIL_POINTER	No axis passed to FB	Ensure function block call only with correct axis passed.
50001	jsmcERR_DRIVE_NOT_READY	controller is not ready to switch on	Check controller for errors
50002	jsmcERR_DRIVE_SWITCHED_OFF	controller is switched off	Don't call function block when controller is switched off
50004	jsmcERR_REFERENCE_WRONG_METHOD	Reference method is not correct for the motor	Check documentation for allowed reference methods for the motor
50006	jsmcERR_ACCE_TO_SMALL	Acceleration is too small	Use larger acceleration ( $\geq 2000$ inc/s <sup>2</sup> )
50008	jsmcERR_SCURVE_NOT_IN_RANGE	Scurve is not in allowed range	Use Scurve in allowed range (1...100%)
50010	jsmcERR_SDO_COMM_FAILURE	Failure during SDO communication	Check power link connection to the Servo Controller
50011	jsmcERR_POWER_UP_FAILURE	Failure during power up sequence	Check Servo Controller for correct power supply
50012	jsmcERR_POWER_LOST	Power was turned off outside of JS_MC_Power control	Check and quit errors from other function blocks or axis, which caused the power off
50013	jsmcERR_WRONG_STATE_FOR_FB	The FB cannot be used in the current state	Check program to call FB's only in allowed states
50014	jsmcERR_WRONG_OP_MODE_FOR_FB	The FB cannot be used in the current mode of operation	Only use allowed FB's for the desired mode of operation (profile position or cyclic synchronized)

Value	Name	Description	Correction
50015	jsmcERR_EXECUTION_ERROR	The FB failed during execution by an external error	Check and quit errors from other function blocks or axes, which caused the fault
50016	jsmcERR_BUFFER_TO_SMALL	The buffer for the error text string is too small	Put a pointer to a buffer for the error text string which size is at least 50 characters
50017	jsmcERR_TEXT_OBJ_NOT_FOUND	Error text object or function block text object not found	Enter correct name of the error text object and ensure, that the error text object (JsMcEtXDe/JsMcEtXEn) and the function block text object (JsMcFBtXEn) are present in the project
50018	jsmcERR_TEXT_READOUT_FAILURE	Error text or function block text could not be read successfully	Ensure that the error text object (JsMcEtXDe/JsMcEtXEn) and the function block text object (JsMcFBtXEn) are present in the project
50019	jsmcERR_WRONG_GENERAL_OP_MODE	general mode of operation not supported	Set a supported general mode of operation in JS_MC_Init (OperationMode = jsmcMODE_PROFILE_POSITION or jsmcMODE_CYCLIC_SYNC)
50020	jsmcERR_REF_SPEED_NOT_IN_RANGE	Reference speed for rotative motors is out of range	Use reference speed in allowed range (0...250000 inc/s)
50021	jsmcERR_ZMARK_SPEED_NOT_IN_RANGE	Z-Mark speed for rotative motors is out of range	Use Z-Mark speed in allowed range (0...100000 inc/s)
50022	jsmcERR_VELOCITY_NOT_IN_RANGE	Velocity is out of range	Use velocity in allowed range (10...9000000 inc/s for linear motor, 10...100000000 inc/s for rotative motor)
50023	jsmcERR_ACCE_TO_LARGE	Acceleration is too large	Use smaller acceleration (smaller than 1000000000 inc/s <sup>2</sup> )
50024	jsmcERR_CYCLE_TIME_FAILURE	Cycle time setting failure	Use correct cycle time setting (bus cycle time >= 200us and software task cycle time >= bus cycle time)
50025	jsmcERR_DECE_TO_SMALL	Deceleration is too small	Use larger deceleration (>=2000 inc/s)
50026	jsmcERR_DECE_TO_LARGE	Deceleration is too large	Use smaller deceleration (smaller than 1000000000 inc/s <sup>2</sup> )
50027	jsmcERR_FW_VERS_FAILURE	Firmware version failure	Use at least XENAX firmware V3.64D
50028	jsmcERR_PDO_MAPPING_CHK_FAILURE	Failure during PDO mapping check	Error in AsIOPVInfo() function block of AsIO library
50029	jsmcERR_PDO_MAPPING_MISSING	Necessary PDO mapping missing	Check, if all necessary PDOs are mapped in I/O Mapping
50030	jsmcERR_NO_DATA_ADDRESS_ASSIGNED	No data address for error text string assigned	Assign valid data address for error text string

Value	Name	Description	Correction
50031	jsmcERR_SDO_ACCESS_FAILURE	Invalid SDO access	Check input values DataObject, SubID and DataLength and set correct values
50032	jsmcERR_CYCLIC_COMM_INTERRUPTED	Cyclic communication interrupted	Don't enable power until JS_MC_CyclicIn is valid and cyclic communication is running
50033	jsmcERR_SPAD_FAILURE	Wrong set point acknowledge setting	
50034	jsmcERR_INDEX_NOTVALID	Index not valid	
50035	jsmcERR_VALUE_OUTOFRANGE	Value not in range	
50036	jsmcERR_FC_INPUTS_NOTVALID	Force calibration inputs not valid	
50037	jsmcERR_FC_NO_LINEAR	Force calibration only with linear motors	
50038	jsmcERR_FC_REF_ERROR	Force calibration: Error during reference	
50039	jsmcERR_FC_MOTION_ERROR	Force calibration: Error during motion	
50040	jsmcERR_UNKNOWN_MOTORTYPE	Unknown motor type	
50041	jsmcERR_VIRTUAL_AXIS_RESET_FAILURE	MC_Reset from Codesys failed, see ErrorID of MC_Reset	

### 3.5 Error sources

The error source block can be found in the ErrorRecord output of the ReadAxisError block. The table below associates sources number with the corresponding function block.

ErrorSource Nr.	Error srouce
1	Axis error or warning
2	CyclicIn
3	Power
4	Reference
5	MoveAbsolute
6	MoveRelative
7	MoveCyclicPosition
8	Stop
9	Halt
10	AxisErrorCollector
11	ReadAxisError
12	ReadParameter
13	WriteParameter
14	JogVelocity
15	ReadActualCurrent
16	ReadDigitalInput
17	ReadDigitalOutput
18	WriteDigitalOuput
19	SetPDO
20	ForceCalibration

### 3.6 Error type

The error type is important for error handling. Because of that, the error type is provided in the error record in an additional field.

ErrorTyp Nr.	ErrorTyp
1	Axis error
2	Axis warnung
3	Function block error

## 4 Example Projects in Codesys

This chapter describes how to put a Jenny Science axis into operation. Example projects are used for this purpose

There are three different examples:

### **Simple Demo**

Axis moves to two alternating positions.

### **Force Limit**

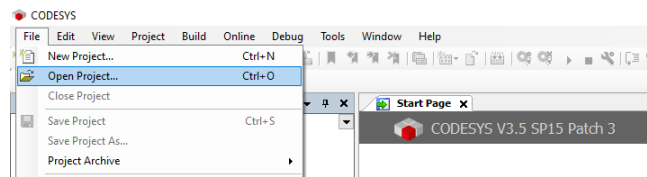
Axis drives forward with a limited Force. If the limited force is reached due to an obstacle in the forward path, the axis stops and moves back quickly.

### **Forceteq**

This is an extended version of the Force Limit example project. This example includes a demo of force monitoring where 3 sectors are defined. When the axis detects an obstacle in the forward path, it will evaluate the sectors and show in which sector the obstacle was.

#### 4.1 Open Project

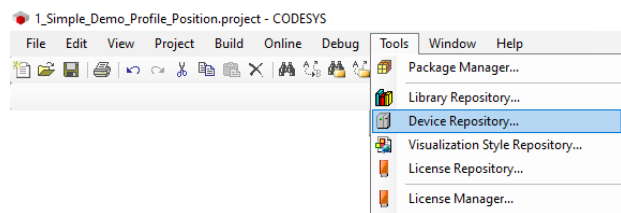
Start Codesys, select “Open Project” and choose one of the demo projects. It is recommended to start with the “Simple Demo” example project.



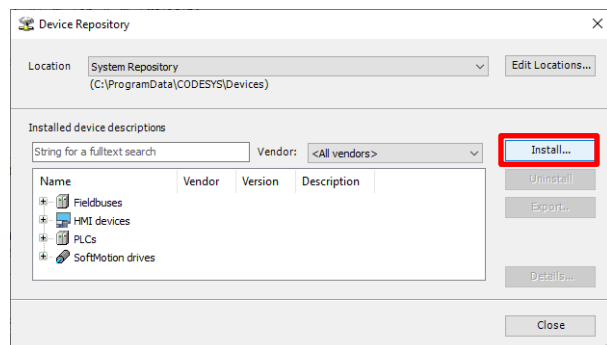
#### 4.2 XENAX® ESI XML Installation

The EtherCAT Slave Information XML for the XENAX can be installed over  
Tool -> Device database

The required XML file can be downloaded from [www.jennyscience.ch](http://www.jennyscience.ch) under  
“XENAX Servocontroller->Firmware Bus Module  
->EtherCAT”.



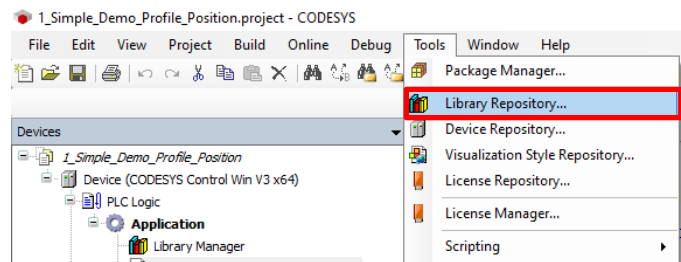
Click add device and select the XML file in the ESI folder. There are other files in the same folder but **only the XML** file is needed.



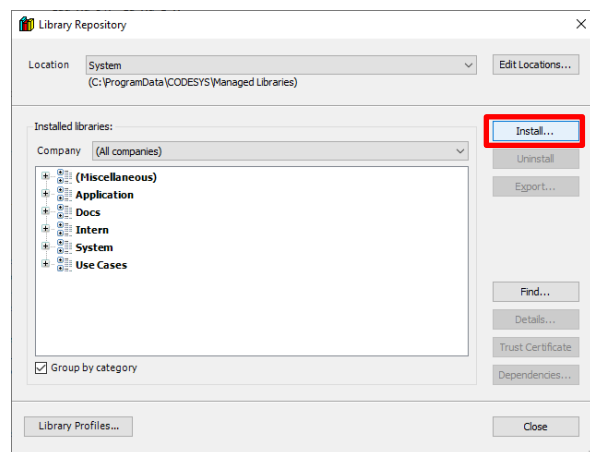


#### 4.3 Library Installation

To install the JS\_MC\_Lib, open the  
“Library repository” over  
Tools->Libraries->Library Repository

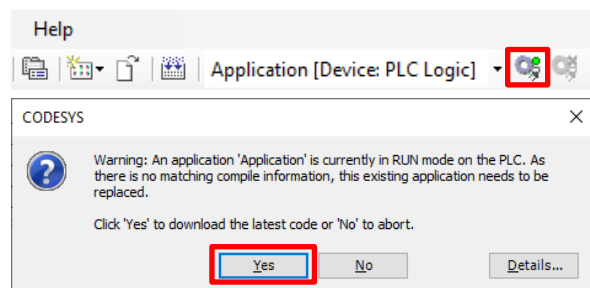


Install the library which is provided in the same  
download as this documentation.



#### 4.4 Download Programm

Go online and download the PLC program.



#### 4.5 Launch Demo Program

Press play to start the demo program.



The following subsection will describe 3 different demo applications in more details.

##### 4.5.1 Simple Demo

This demo initialises and powers on the motor.  
After that, the demo programs calls MC\_MoveAbsolute with an alternating target position of 0 and 44'000 increments.

##### 4.5.2 Force Limit

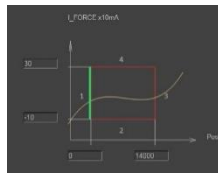
This project demonstrates the force limitation part of Forceteq®. The axis drives forward with a limited force. If an obstacle is in the forward path, the force limit will be reached and the axis moves back quickly to the starting position.

After power up, a Force Calibration is performed. During a Force Calibration, the motor drives slowly from start- to end position and records friction and cogging. Those forces are then automatically compensated by the XENAX®. Such a calibration makes the force limitation much more accurate.

##### 4.5.3 Force Sectors

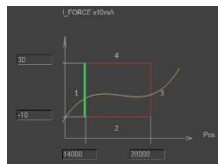
This demo expands the "Force Limit" demo program with the features of Force Monitoring. With Force Monitoring, it is possible to define force-position sectors. After a drive is finished, each sector is either in a successful or failed state. Each edge of a sector can be defined as entry or exit edge or both. A sector is successful if the curve entered through an entry edge and left through an exit edge. The demo program defines 3 such sectors without an exit edge. This means that a sector is successful if the curve does not leave the sector. This happens in the demo program if an obstacle was detected in that sector.

Force-position sectors configured by PLC software and viewed in Webmotion:



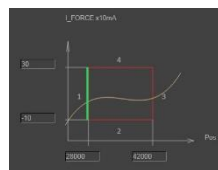
\*\*\*\*\* Sector I\_Force 1 \*\*\*\*\*

Sector IForce Start = 0  
Sector IForce End = 14000  
IForce Low x10mA = -10  
IForce High x10mA = 30  
Sector Transit Config = 4096



\*\*\*\*\* Sector I\_Force 2 \*\*\*\*\*

Sector IForce Start = 14000  
Sector IForce End = 28000  
IForce Low x10mA = -10  
IForce High x10mA = 30  
Sector Transit Config = 4096

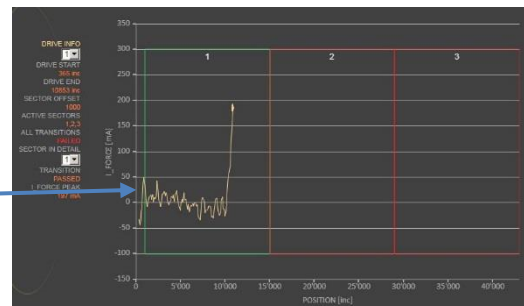


\*\*\*\*\* Sector I\_Force 3 \*\*\*\*\*

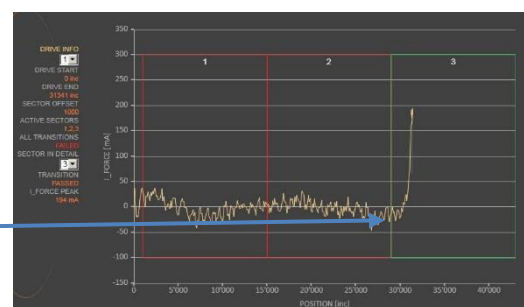
Sector IForce Start = 28000  
Sector IForce End = 42000  
IForce Low x10mA = -10  
IForce High x10mA = 30  
Sector Transit Config = 4096

The variables of the main program indicate in which sector the obstacle was detected.

Expression	Type	Value
State	INT	2022
bInitDoneBB	BOOL	TRUE
bForceCalibDone	BOOL	TRUE
StartPosition	DINT	0
EndPosition	DINT	44000
IForce_Test	UDINT	20
FT_State	INT	99
WaitTime	INT	0
TouchSector1	BOOL	TRUE
TouchSector2	BOOL	FALSE
TouchSector3	BOOL	FALSE
NoTouch	BOOL	FALSE
TestError	BOOL	FALSE



Expression	Type	Value
State	INT	2022
bInitDoneBB	BOOL	TRUE
bForceCalibDone	BOOL	TRUE
StartPosition	DINT	0
EndPosition	DINT	44000
IForce_Test	UDINT	20
FT_State	INT	99
WaitTime	INT	0
TouchSector1	BOOL	FALSE
TouchSector2	BOOL	FALSE
TouchSector3	BOOL	TRUE
NoTouch	BOOL	FALSE
TestError	BOOL	FALSE
_IoMap	JS_MC_IoMap	

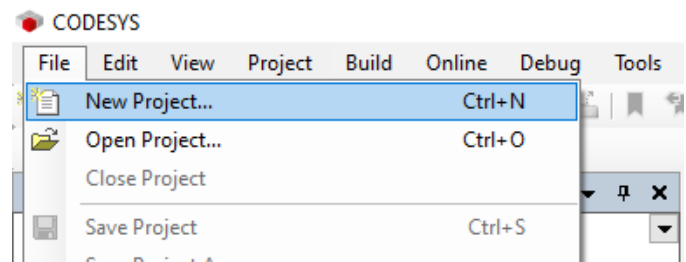


## 5 New Project profile position in Codesys

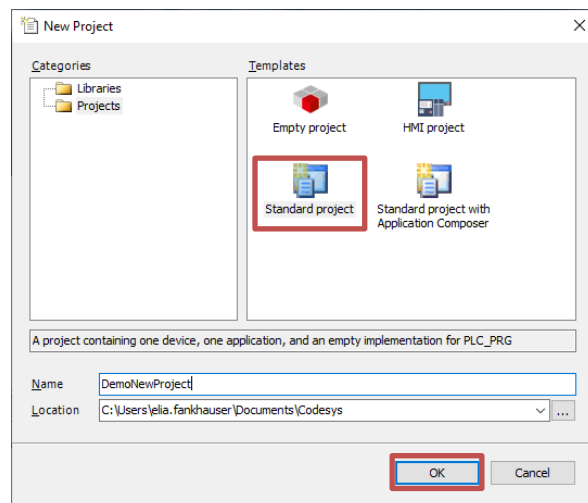
This chapter describes how to put a Jenny Science Axis into operation without a demo project. It is possible to create a new project or to add a Jenny Science axis into an existing project.

### 5.1 Create Project

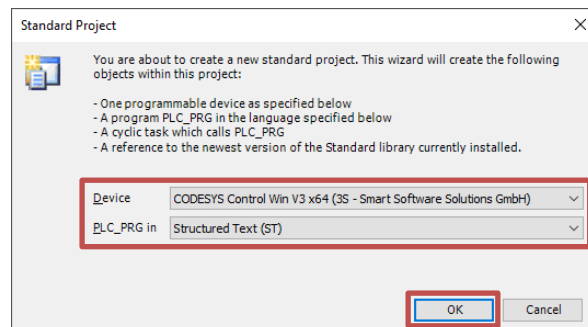
File→New→Project...



Select Empty project and click ok.

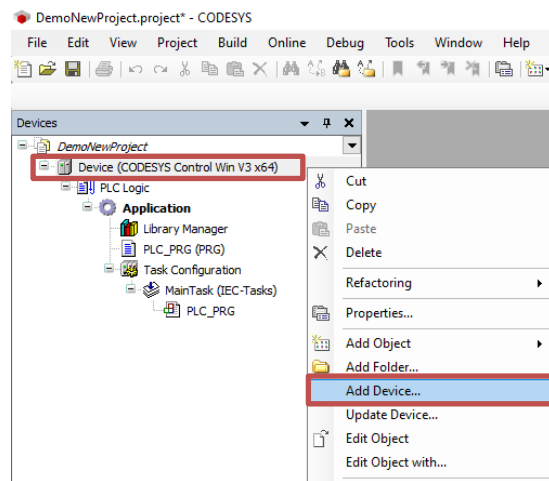


Select your device and your preferred programming language. The device "CODESYS Control Win V3 x64" can be used to simulate a PLC on the local windows machine.

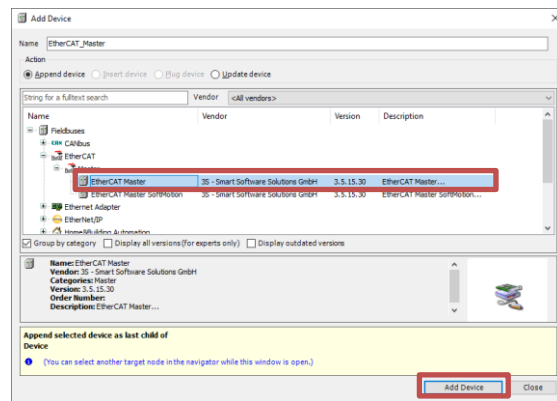


## 5.2 Add EtherCAT Master

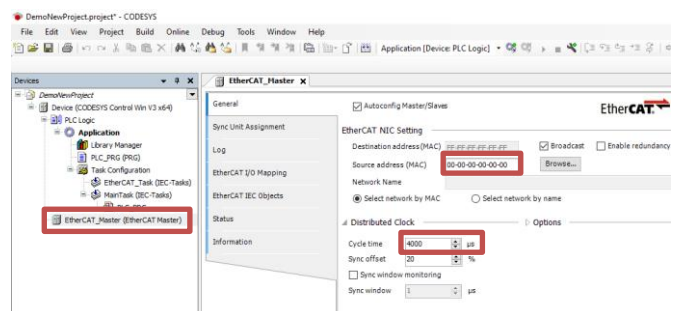
Add an EtherCAT master under Add Device.



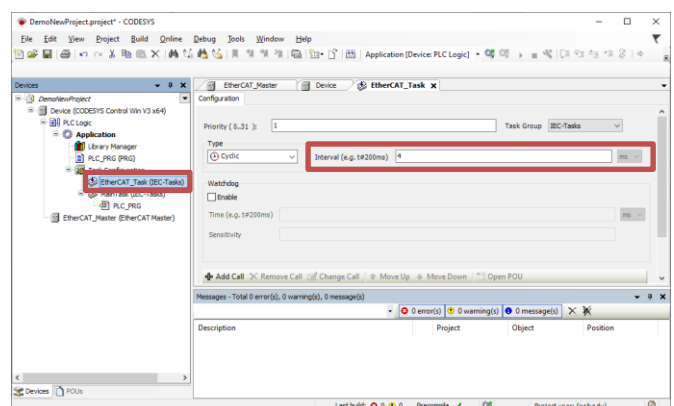
Select EtherCAT master and add the device.



Select the network interface which shall be used for EtherCAT communication and define the bus cycle time

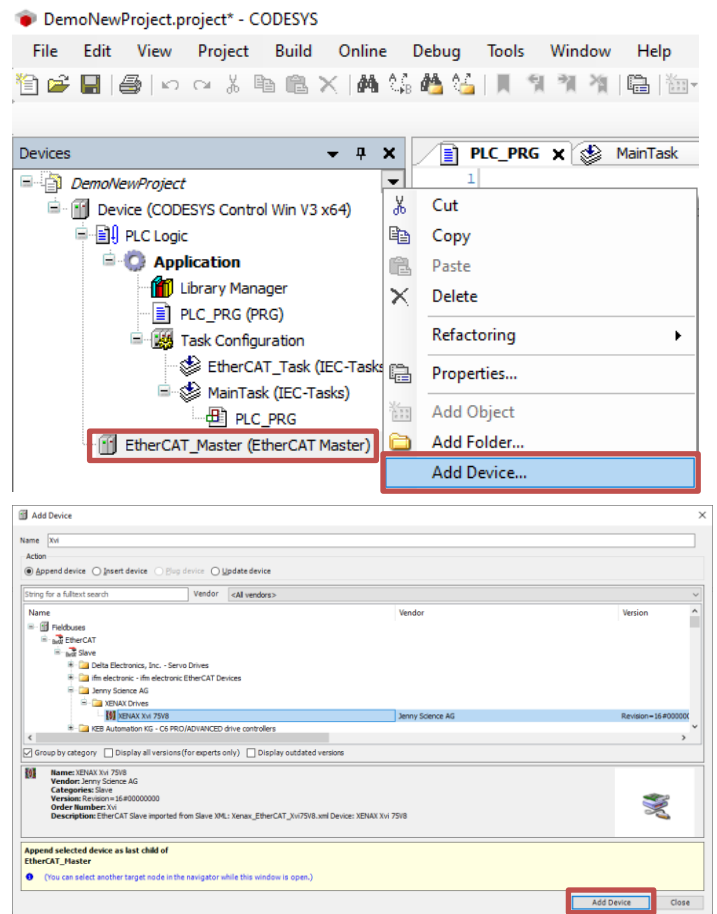


The Interval of the EtherCAT task must be set to the same value as the bus cycle time.



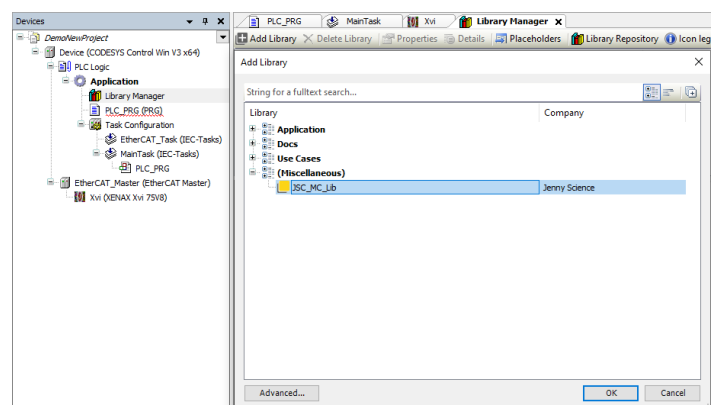
## 5.3 Add XENAX®

Add the XENAX® servo controller as a slave to the EtherCAT bus. Note that the XENAX® must be installed first as described in section 4.2 XENAX® ESI XML Installation.



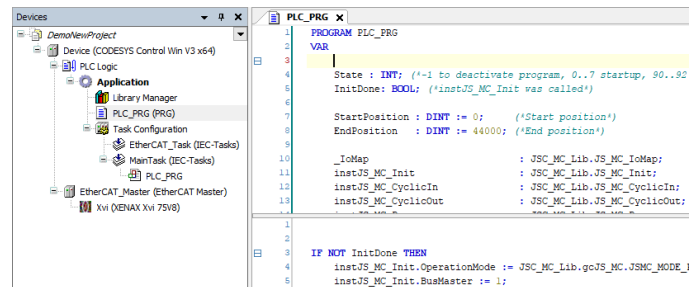
## 5.1 Add Jenny Science Library

Add the installed JSC\_MC\_Lib.  
Installing the JSC\_MC\_Lib is described in 4.3 Library Installation.



## 5.2 Copy Code

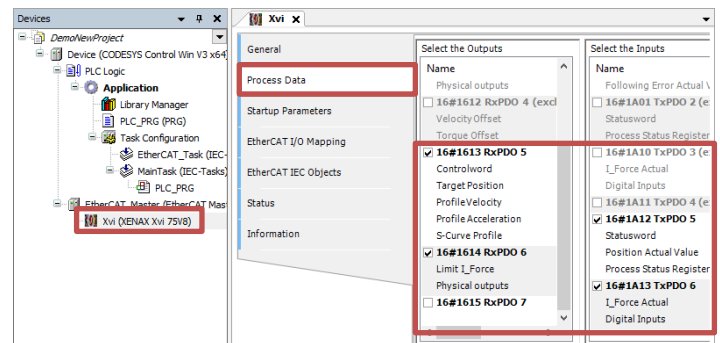
Copy the code from a demo application.



## 5.3 PDO Mapping

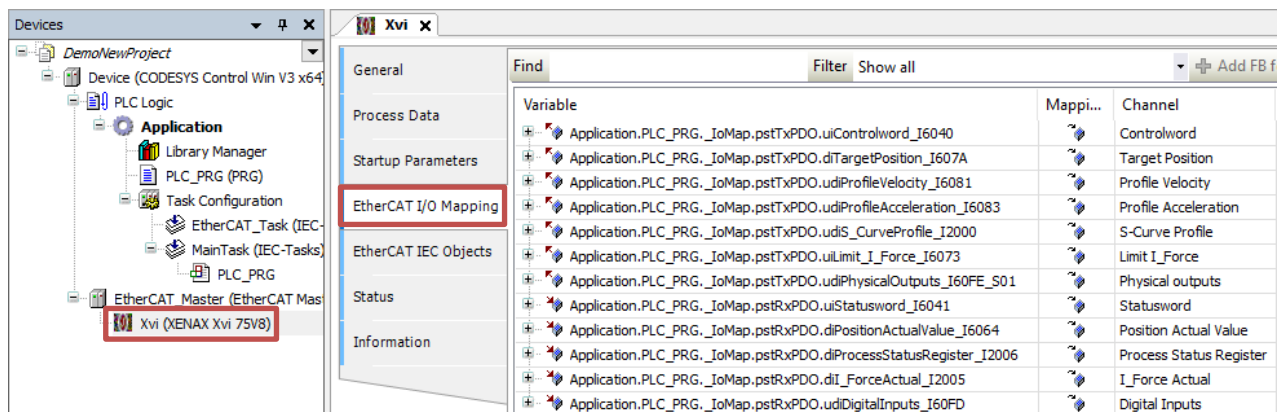
Enable the two required PDOs. There are two additional PDOs which can be enabled for additional features.

Rx	Tx
<b>required PDOs</b>	
PDO 5	PDO 5
<b>Optional PDOs</b>	
PDO 6	PDO 6



## 5.4 PDO Linking

Map all PDO channels to a variable in the \_IOMap structure.



## 5.5 Launch Project

Press play to compile and download the project.  
Make sure there are no errors.

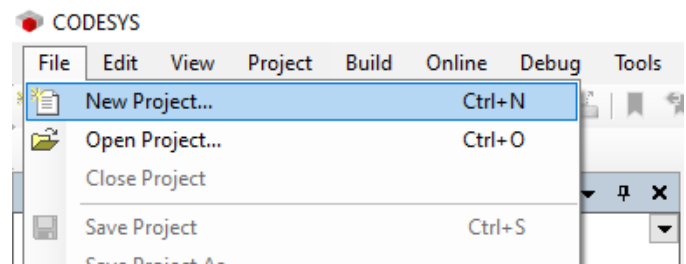


## 6 New Project Softmotion CSP in Codesys

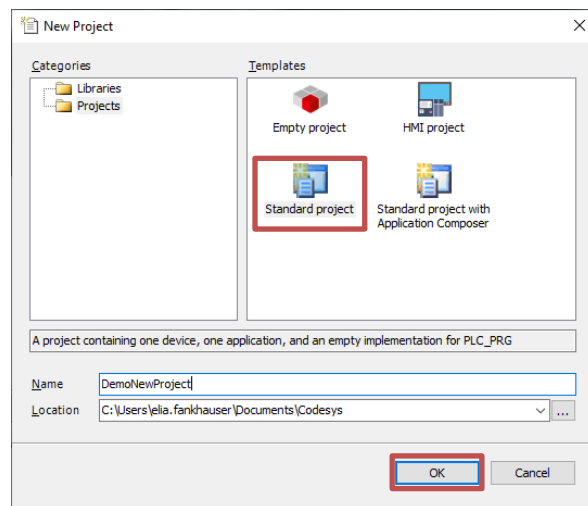
This chapter describes how to put a Jenny Science Axis into operation without a demo project. It is possible to create a new project or to add a Jenny Science axis into an existing project.

### 6.1 Create Project

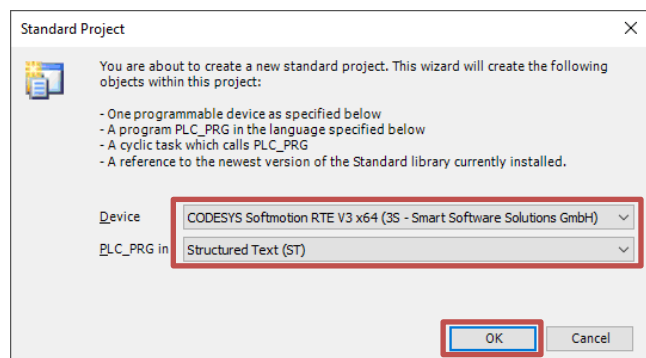
File→New→Project...



Select Empty project and click ok.



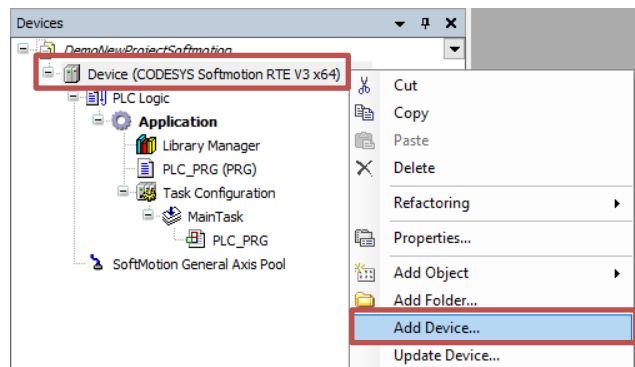
Select your device and your preferred programming language. The device “CODESYS Softmotion RTE V3 x64” can be used to simulate a PLC on the local windows machine.



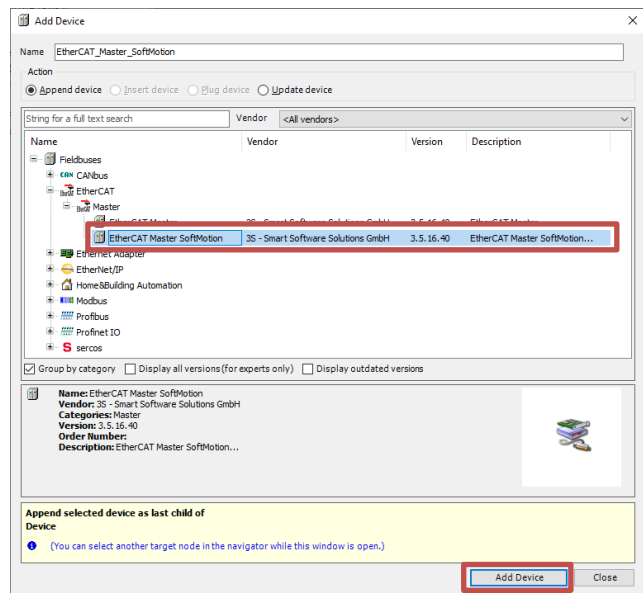


## 6.2 Add EtherCAT Master

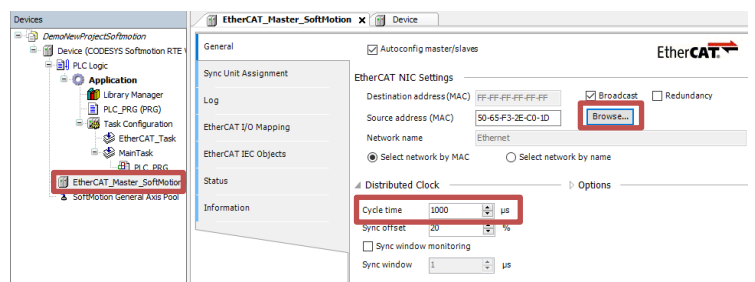
Select add Device on the PLC device to add a EtherCAT Master.



Select the EtherCAT Master Softmotion and click on add device.

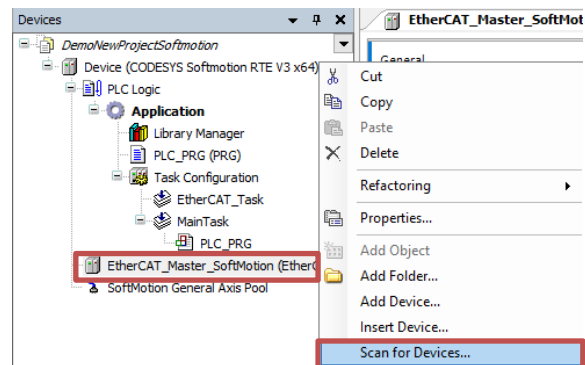


Reconfigure the bus cycle time to at least 1ms and make sure that the correct MAC address is selected.

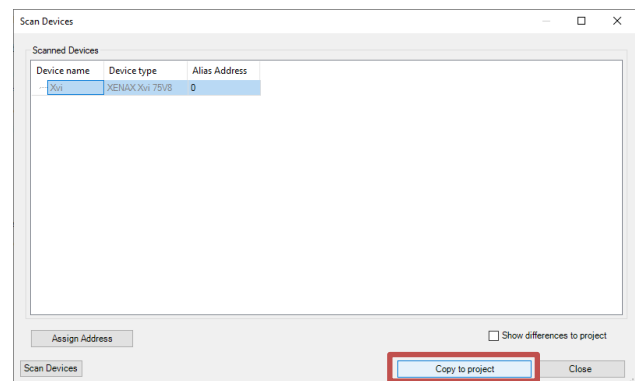


### 6.3 Add XENAX® as EtherCAT Slave

The XENAX® can be added as a slave device by scanning on the EtherCAT master.

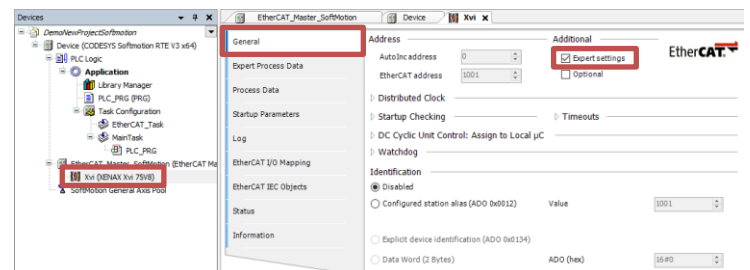


Copy the XENAX® into the current project.

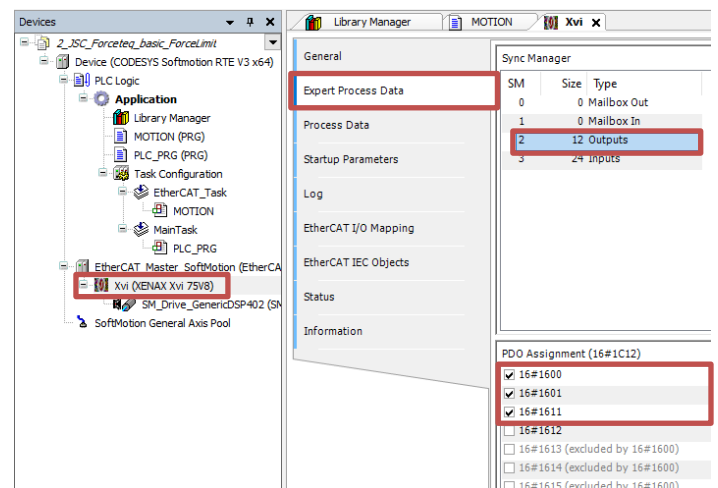


### 6.4 Configure XENAX®

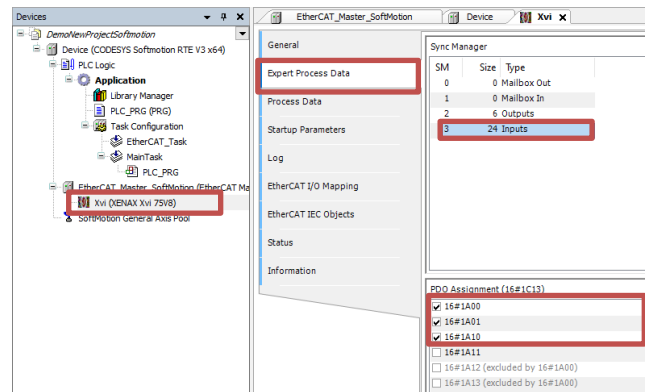
Check expert settings to unlock all options.



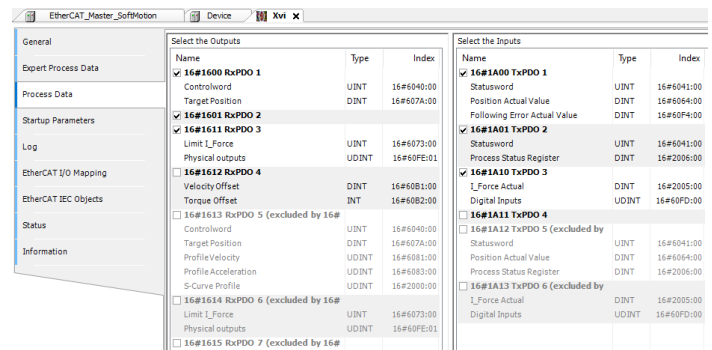
In the Expert Process Data settings, enable the first three PDOs



Also enable the first three input PDOs.

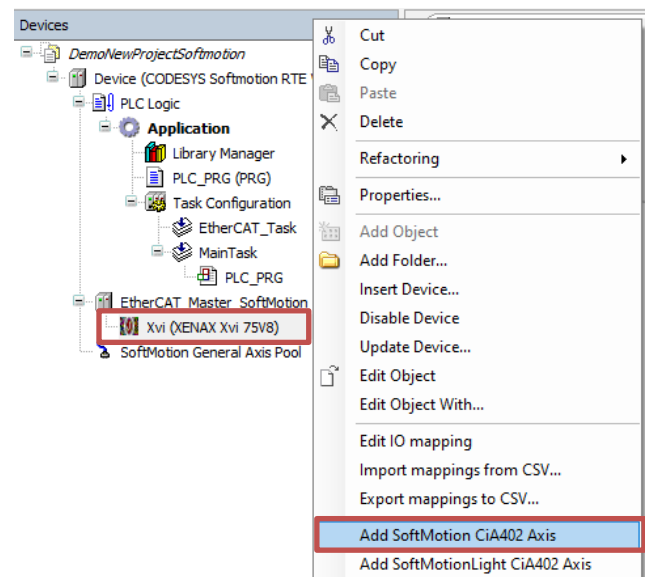


The process data section should now look like this.

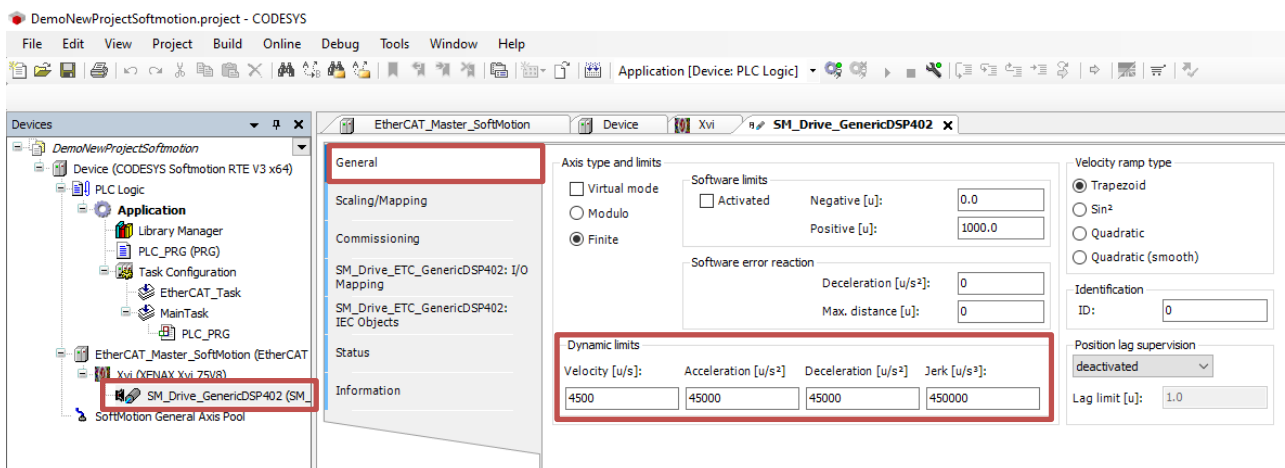
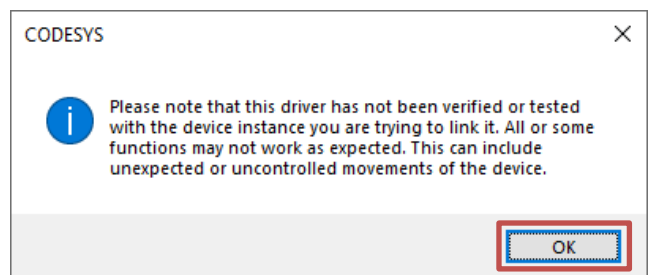


## 6.5 Add Softmotion Axis

Add a Softmotion axis below the XENAX® Servocontroller.



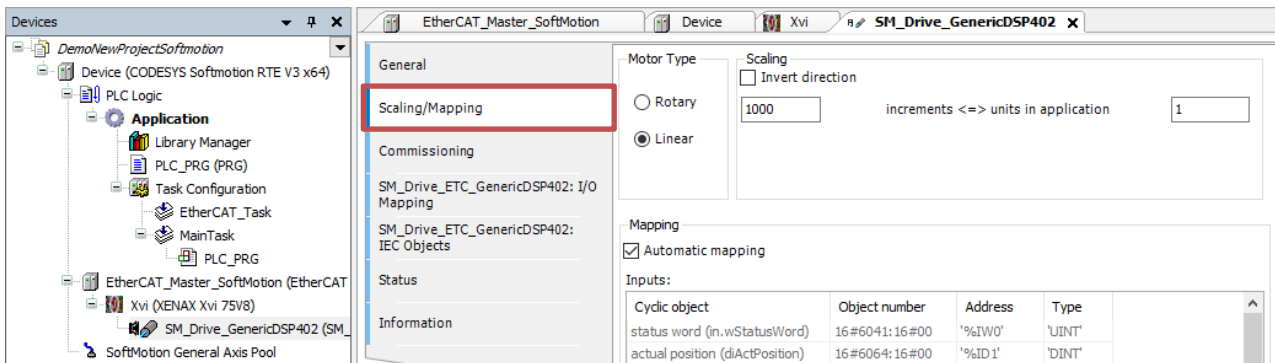
Press ok.



JSC Motor Type	Resolution Scale	Velocity	Acceleration	Deceleration	Jerk
LINAX®, ELAX®	1 µm/inc	4500	45000	45000	450000
LINAX®	100 nm/inc	900	9000	9000	90000
ROTAX® Rxvp	0.005625 deg/inc	9000	90000	90000	900000
ROTAX® Rxhq	0.003 deg/inc	7200	72000	72000	720000

Configure the Dynamic limits section according to the motor type.

The scaling depends on the motor type too. The suggested scaling below is made so that a unit for linear motors is 1mm and a unit for rotary motors is 1° degree.



LINAX® or ELAX with resolution scale of 1µm/inc:

Motor Type	Scaling
<input type="radio"/> Rotary	<input type="checkbox"/> Invert direction
<input checked="" type="radio"/> Linear	1000 increments <=> units in application 1

LINAX® with resolution scale of 100nm/inc:

Motor Type	Scaling
<input type="radio"/> Rotary	<input type="checkbox"/> Invert direction
<input checked="" type="radio"/> Linear	10000 increments <=> units in application 1

ROTAX® Rxvp:

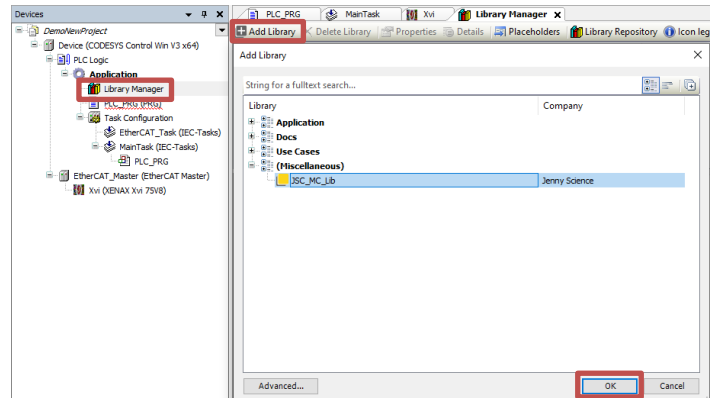
Motor Type	Scaling
<input checked="" type="radio"/> Rotary	<input type="checkbox"/> Invert direction
<input type="radio"/> Linear	64000 increments <=> motor turns 1
	1 motor turns <=> gear output turns 1
	1 gear output turns <=> units in application 360

ROTAX® Rxhq:

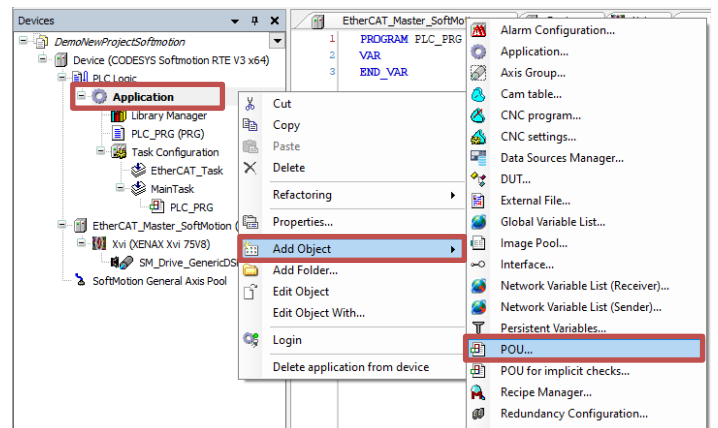
Motor Type	Scaling
<input checked="" type="radio"/> Rotary	<input type="checkbox"/> Invert direction
<input type="radio"/> Linear	120000 increments <=> motor turns 1
	1 motor turns <=> gear output turns 1
	1 gear output turns <=> units in application 360

## 6.6 Add Program Code

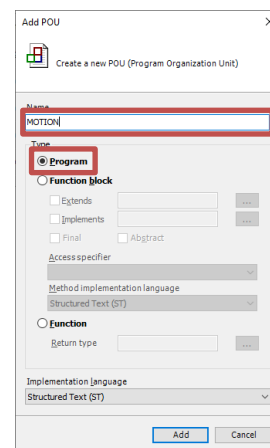
Add the installed JSC\_MC\_Lib.  
Installing the JSC\_MC\_Lib is described in 4.3  
Library Installation.



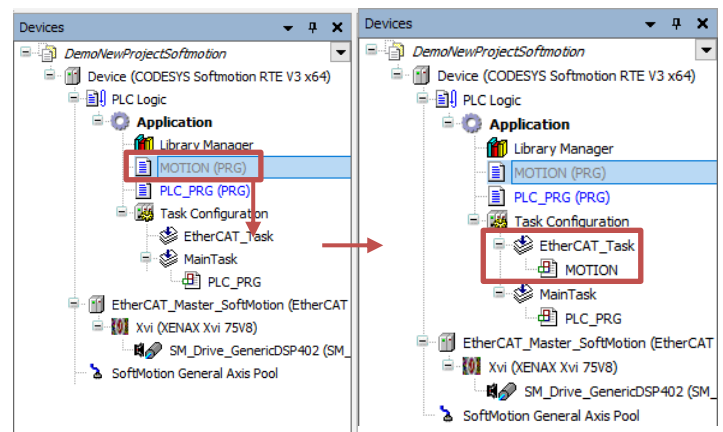
All Softmotion commands must be called from  
the EtherCAT task. Add a new POU under  
Application.



Give it a name and select Programm.

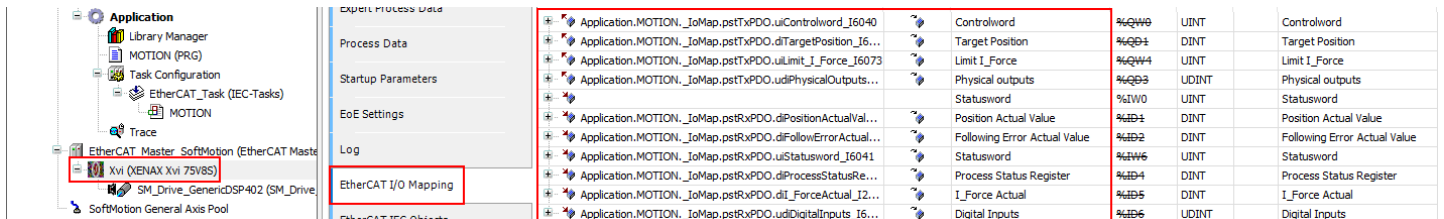


Drag and Drop the new Motion programm to the  
EtherCAT task.



## 6.7 PDO Linking

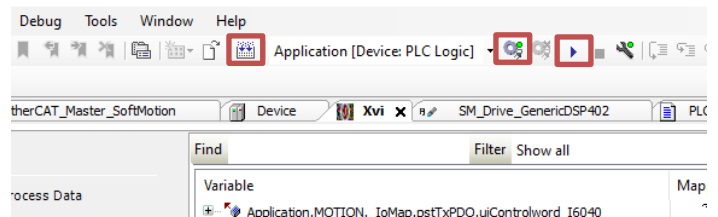
Some PDOs need to be linked with a variable.  
The JSC\_MC\_Library has a IoMap structure  
intended for linking variables with PDOs.



Variable	Variable Type	Variable Name
Controlword	UINT	Controlword
Target Position	DINT	Target Position
Limit I_Force	UINT	Limit I_Force
Physical outputs	UDINT	Physical outputs
Statusword	UINT	Statusword
Position Actual Value	DINT	Position Actual Value
Following Error Actual Value	DINT	Following Error Actual Value
Statusword	UINT	Statusword
Process Status Register	DINT	Process Status Register
I_Force Actual	DINT	I_Force Actual
Digital Inputs	UDINT	Digital Inputs

## 6.8 Download the Project

Compile, download and run the project.



## 7 Version history

Version	Date	Changelog
1.1.6	19.07.2024	- reset following error is performed faster. This function is used in JS_MC_STOP, JS_MC_Reference and JS_MC_ForceCalibration.
1.1.5	09.07.2024	- fixed JS_MC_Power stuck in enPOWER_W4_SHUTDOWN when Enable input was removed during a reference drive - fixed JS_MC_Reference stayed busy after it was aborted by power loss or SMU error
1.1.4	23.05.2024	- csp only: JS_MC_Power could get stuck in enPOWER_CYC_POWERON when enable input was removed during power on.
1.1.3	07.05.2024	- csp only: Axis PDO Linking changed, see 6.7 - support for Intax and Lxs absolute - new reference mode "do not change" which uses the reference settings configured in Webmotion - fixed a bug which caused an error in the Codesys virtual axis after reference or force calibration
1.1.2	15.03.2024	- JS_MC_Init waits until slave address is not 0, this is the case in the first program cycle - JS_MC_Init can be called during normal operation to update pointers which may change during online changes
1.1.1	14.03.2024	- csp only: JS_MC_Init requires a pointer to the Codesys virtual axis instead of just the slave address - faster axis disconnection recognition - additional signal IsOperational in JS_MC_CyclicIn for users to reconnect with a disconnected axis
1.1.0	27.05.2021	- added support for cyclic synchronous position mode(csp) - renamed a few function blocks for better readability
1.0.0	14.02.2020	- Initial release with support for profile position



This instruction manual contains copyright protected information. All rights are reserved. This document may not be entirely or partially copied, duplicated or translated without the prior consent of Jenny Science AG.

Jenny Science AG grants no guarantee on, or will be held responsible for, any incidents resulting from false information.

Information in this instruction manual might be subject to change.

Jenny Science AG  
Sandblatte 11  
CH-6026 Rain

Tel +41 (0) 41 255 25 25

[www.jennyscience.ch](http://www.jennyscience.ch)  
[info@jennyscience.ch](mailto:info@jennyscience.ch)

© Copyright Jenny Science AG 2024