Date: 2021-07-16	Author: HC	Name: FileName.docx	
Version: 0.1	Reviewer:	Approver:	**Document** IOLINK Manual PRELIMINARY

Table of contents

1. History of Document

Version	Review status	Date	Ini
0.1	First edition	2021-07-16	HC

1.1 Heading

The project plan states that:
"JVL A/S has no documented skills within the topics of Functional Safety. A certification (FSCED) of 1 or more cooperators of JVL will be agreed (the FSM as a minimum) with TÜV NORD during the assessment of this project."
A certificate is missing, but the FSCED workshop is planned for 2021 or whenever it is possible.

2. JVL IOLINK Gateway

2.1 Add On Instructions

From the gateway an RS422 Modbus connection goes to a MIS motor. Using a Q9 motor type the Modbus connection can be multidropped to several other motors.

The firmware supports different Modbus function codes.
The standard modbus 0x3 and 0x10 (Read, Write Coil), but also more data optimized JVL specific codes for PDO like data transmission.
The special PDO frames has the function codes:
0x4A and 0x4B
The IOLINK connection is limited to 32bytes each way, and the gateway is 100% transparent, meaning that the PLC running the IOLINK master will form the Modbus package that goes to the motor. The package is then continuously transmitted (back to back) on the RS422 line.
So out of the 32bytes 23bytes are used to transfer a Modbus package (PDO) with values. Another 23bytes are received when the motor responds.
So for the PDO transmissions a TX PDO with payload of 5 registers (5x4 bytes)is sent and as a response a Rx PDO is received from the motor with data from another 5 registers.
To have enough data 2x Rx and Tx PDO's are configured 10 registers each way in total.
For the Rockwell example the following is configured:

2.2 Tx PDO1:

2 – Mode register
3 – Requested position
5 – Requested velocity
6 – Acceleration
7 – Torque (Current)

2.3 Rx PDO1:

2 – Mode
10 – Actual position
12 – Actual velocity
212 – Actual torque (only H2/H4 encoder equipped motors)
25 – Status bits (Inposition, accelerating, decelerating, homing done etc)

2.4 Tx PDO2:

38 – Homing position
40 – Homing velocity
41 – Homing Torque
242 – Homing crawl velocity
144 – P_NEW (used for passive homing and relative motion)

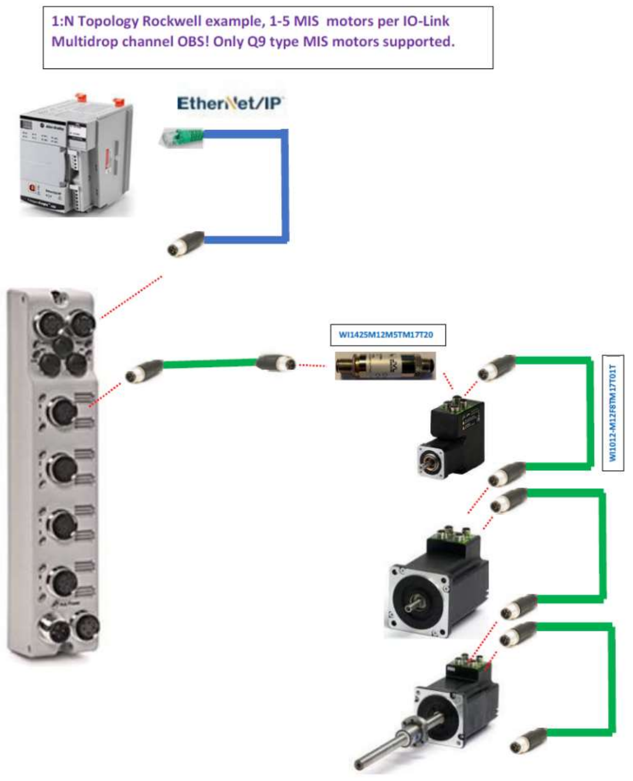
2.5 Rx PDO2:

35 – Error
36 - Warnings
246 – Temperature, high resolution
143 – CVI control voltage
19 - Inputs

The IOLINK integration in Rockwell is based on the existing AOI's for the MIS's. Obviously They are developed for running under EthernetIP.
From EthernetIP 8x read and 8x write registers are cyclic exchanged with each motor. The data exchanged and the internal scaling is handled by a COM-AOI, which also handles the different modes in the motor including homing.
The IOLINK application isn't much different except that the Modbus frames need to be configured and addressed to each motor.
All that is handled in a dedicated COM -AOI, but in contrast to the one used for EthernetIP, the one for IOLINK supports more motors. In order to have an acceptable performance not more than 5 motors should be controlled at the same time.
So, for each IOLINK channel from the master up to 5 motors can be controlled, provided that the RPI for the IOLINK master is kept as low as possible.
Timing performance has not yet been assessed nor optimized.
This concept will be considerably slower than running EthernetIP, however in many applications this is tolerable.

The RS422 Modbus channel requires that each motor is configured with a static motor address starting from addr = 1 up to 5. As an input parameter the amount of motors on the RS422 line is entered in the "Motors" parameter of the IOLNK_Com -AOI.
The IOLNK_COM AOI will handle the cyclic data exchange to and from all the motors.

The Data flow is as stated below:


1:N Topology Rockwell example, 1-5 MIS motors per IO-Link Multidrop channel OBS! Only Q9 type MIS motors supported.
EtherNet/IP
WI1425M12M5TM17T20
WI1012-M12F8TM17T01T

Data is formed by the IOLNK_COM and placed in the Cyclic data array from the PLC to the IOLINK master module, which runs EthernetIP

↓ EthernetIP

Data is transferred from the IOLINK Master to the JVL gateway using IOLKINK on COM2 (460K)

↓ IOLINK

The JVL gateway received the data and transparently sends the data out on the RS422 line.

↓ RS422, Modbus

The motor(s) receives the data and send back a response which can be a PDO or a normal ACK if rewad/write coil is used as function codes.

3. IOLINK gateway communication

The JVL iolink adapters is basically a gateway between the IOLINK master and the JVL motors converting IOLINK to a Modbus stream.

The IOLINK payload consists of 32bytes that is transferred each way.
The Modbus telegram is embedded into the 32bytes frames. Some internal special non modbus compliant commands enables PDO of up to 7 words each way. But standard modbus commands 0x3 and 0x10 are supported, so single registers in the motor can be set and read.

Since multidrop topology is supported the modbus line and iolink channel are shared between the motors so optimization of the bandwidth is needed to get the best cycle time possible. The motor firmware supports configuration of 2 individual RX and TX PDO (Process Data Object), that will optimize the bandwidth of the Rs422 modbus channel

Each motor on the RS422 line has an address assigned (motor address) so each motor can be accessed individually.

3.1 Read a register (0x3 read coil)

Reading a register from a specific motor requires the following byte mapping.

Byte index	Value	Description
0	Motor address	The configured motor address
1	0x3	Read Coil Command
2	Register_HI	Register number to read from the motor (x2, 16bit representation) Hi word
3	Register_LO	Register number to read from the motor (x2, 16bit representation) Hi word
4	N_Bytes_Hi	Amount of bytes to read, Hi value
5	N_Bytes_Lo	Amount of bytes to read, Lo

Example:

Read the actual position (Register 10) from motor with address 4.

Byte index	Value	Description
0	4	The configured motor address
1	0x3	Read Holding register Command
2	0	Register number to read from (Start address) the motor (x2, 16bit representation) Hi word
3	20	Register number to read from (Start address) the motor (x2, 16bit representation) Lo word
4	0	Amount of bytes to read, Hi value
5	4	Amount of bytes to read, Lo

Lets assume the motor is located in position = 100000 counts (= 0x000186A0)

Motor response:

Byte index	Value	Description
0	4	The configured motor address
1	0x3	Read Holding register Command
2	4	Amount of bytes
3	0x86	Value[1]
4	0xA0	Value[0]
5	0x00	Value[3]
6	0x01	Value[2]

3.2 Write a value to a register (0x10 write coil)

Writing a register requires the following mapping:

Byte index	Value	Description
0	Motor address	The configured motor address
1	0x10	Write multiple registers
2	Register_HI	Register number to read from the motor (x2, 16bit representation) Hi word
3	Register_LO	Register number to read from the motor (x2, 16bit representation) Hi word
4	N_Bytes_Hi	Amount of bytes to read, Hi value
5	N_Bytes_Lo	Amount of bytes to read, Lo

3.3 Optimized cyclic PDO data exchange

3.4 PDO Configuration

When working with a normal Modbus telegram, the initiator sends a request and the consumer will send back an acknowledge.

So lets assume that we want to set the requested position (register 6) in the motor with address = 4.

We send:

0x4 0x10 0x0 0x6 0x0 0x4 xx xx xx xx (note that crc is taken care of elsewhere)

Assumed that everything goes well, the motor will respond with:

0x4 0x10 0x0 0x6 0x0 0x4

To optimize the use of the limited bandwith a special set of Modbus function codes has been developed.

0xF2 and 0xF3 that combines both reading and writing in one frame. The PDO's are configured to hold the register of interest and each time a write PDO function code is sent, the corresponding read PDO is returned as acknowledge to the request.

Exc. Lets assume that we have configured the Tx PDO1 to hold the following registers:

Tx PDO Index	Register number in the motor	Description
0	2	Mode register. 0=Passive, 1=velocity, 2=position
1	3	Requested position
2	5	Requested velocity
3	6	Requested acceleration
4	7	Requested torque (Current, 1533 = Max. Current)

And the Rx PDO1 to hold the following setup:

Rx PDO Index	Register number in the motor	Description
0	2	Mode register. 0=Passive, 1=velocity, 2=position
1	10	Actual position
2	12	Actual velocity
3	25	Status bits
4	214	Actual torque, for motors supporting closed loop current control.

Everytime a frame is sent to set the registers configured in the Tx PDO, a frame holding data from the registers configured in the Rx PDO, will be returned.

For the Tx PDO1 the Modbus function code is 0x4A and a frame could look like this:

Sending the Tx PDO data to motor addr=4:

Mode = 2
Requested position = 10000 (0x2710)
Requested velocity = 20000 (0x4E20)
Requested Acceleration = 1000 (0x3E8)
Requested torque = 1.00A (1A = 511 = 0x1FF)

0	4	The configured motor address
1	0x4A	PDO1 Function code
2	0	Mode = 2, Byte 1
3	2	Byte 0
4	0	Byte 3
5	0	Byte 2
6	0x27	Requested position = 0x2710, byte 1
7	0x10	Byte 0
8	0x0	Byte 3
9	0x0	Byte 2
10	0x4E	Requested Velocity = 0x4E20, Byte 1
11	0x20	Byte 0
12	0x0	Byte 3
13	0x0	Byte 2
14	0x3	Requested Acceleration = 0x3E8, Byte 1
15	0xE8	Byte 0
16	0x0	Byte 3
17	0x0	Byte 2
18	0x1	Requested Torque = 0x1FF, Byte 1
19	0xFF	Byte 1
20	0x0	Byte 3
21	0x0	Byte 2

The motor will acknowledge the frame with the Rx PDO:

The values as read from MacTalk
Mode = 2
Actual position = 409600 (0x64000)
Actual velocity = 9980 (0x26FC)
Status bits = 0x8A474810
Actual torque = 425 (0x1A9, Appx. 20%)

0	4	The configured motor address
1	0x4A	PDO1 Function code
2	20	Byte count hi
3	0	Mode, Byte 1
4	2	Byte 0
5	0	Byte 3
6	0	Byte 2
7	0x40	Actual position, Byte 1
8	0x0	Byte 0
9	0x0	Byte 3
10	0x6	Byte 2
11	0x26	Actual velocity, Byte 1
12	0xFC	Byte 0
13	0x0	Byte 3
14	0x0	Byte 2
15	0x48	Status bits, Byte 1
16	0x10	Byte 0
17	0x8A	Byte 3
18	0x47	Byte 2
19	0x1	Actual torque, Byte 1
20	0xA9	Byte 0
21	0x0	Byte 3
22	0x0	Byte 2

The PDO's are configured by writing the settings to the following registers:

0xF2	Tx PDO 1
0xF3	Rx PDO 1
0xF5	Tx PDO 2
0xF6	Rx PDO 2

So to use PDO 1 as illustrated in the previous example where the PDO was configured as follows:

Rx PDO Index	Register number in the motor	Description
0	2	Mode register. 0=Passive, 1=velocity, 2=position
1	10	Actual position
2	12	Actual velocity
3	25	Status bits
4	214	Actual torque, for motors supporting closed loop current control.

The write register frame needs to be configured as follows:

0	4	The configured motor address
1	0x10	Write coil modbus function
2	0xF3	Address Lo of the RX PDO1 configuration
3	0	Address Hi
4	0	Words to write (Hi)
5	10	Words to write (Lo)
6	20	Bytes to write
7	0	Mode, Byte 1
8	2	Byte 0
9	0	Byte 3
10	0	Byte 2
11	0	Actual position, Byte 1
12	10	Byte 0
13	0	Byte 3
14	0	Byte 2
15	0x0	Actual velocity, Byte 1
16	12	Byte 0
17	0	Byte 3
18	0	Byte 2
19	0	Status bits, Byte 1
20	25	Byte 0
21	0x0	Byte 3
22	0x0	Byte 2
23	0	Actual torque, Byte 1
24	214	Byte 0
25	0	Byte 3
26	0	Byte 2

Now the Rx PDO 1 has been configured and next step is to configure the Tx PDO 1.

Tx PDO Index	Register number in the motor	Description
0	2	Mode register. 0=Passive, 1=velocity, 2=position
1	3	Requested position
2	5	Requested velocity
3	6	Requested acceleration
4	7	Requested torque (Current, 1533 = Max. Current)

The frame will be configured as follows:

0	4	The configured motor address
1	0x10	Write coil modbus function
2	0xF2	Address Lo of the TX PDO1 configuration
3	0	Address Hi
4	0	Words to write (Hi)
5	10	Words to write (Lo)
6	20	Bytes to write
7	0	Mode, Byte 1
8	2	Byte 0
9	0	Byte 3
10	0	Byte 2
11	0	Requested position, Byte 1
12	3	Byte 0
13	0	Byte 3
14	0	Byte 2
15	0x0	Requested velocity, Byte 1
16	5	Byte 0
17	0	Byte 3
18	0	Byte 2
19	0	Requested acceleration, Byte 1
20	6	Byte 0
21	0	Byte 3
22	0	Byte 2
23	0	Requested torque, Byte 1
24	7	Byte 0
25	0	Byte 3
26	0	Byte 2

The Rx and TX -PDO2 are configured in the same way, just using the function codes 0xF5 and 0xF6.

Please note that using the current motor firmware V5.04 the PDO configuration needs to be done at every startup. The settings are not saved permanently in the motor. The settings of the PDO's might be done automatically in future firmware version.

5. Rockwell Add On Instructions, IOLINK

A series of Add On Instructions (AOI) helps controlling the motors in a defined way. The AOI's are exactly the same whether EthernetIP or IOLINK is used a communication for controlling the motors.

Common for both bus technologies is that a dedicated AOI handles the data to and from the motors as well as the internal register settings during the different operating states of the motor.

There are 2 dedicated communication AOI's one for EthernetIP and one for IOLink.

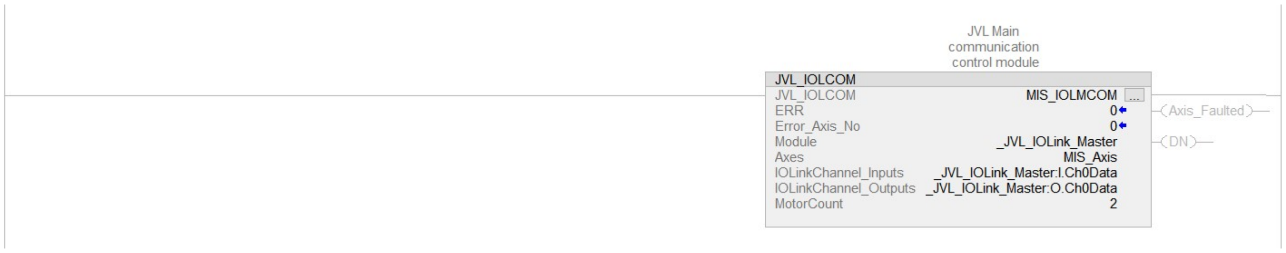
5.1 JVL_IOLCOM, IOLink communication handler

In contrast to the JVL_COM EthenetIP communication handler, the JVL_IOLCOM can handle communication with more than one motor. The limitation is 5 motors on a single JVL_IOLCOM –instance.

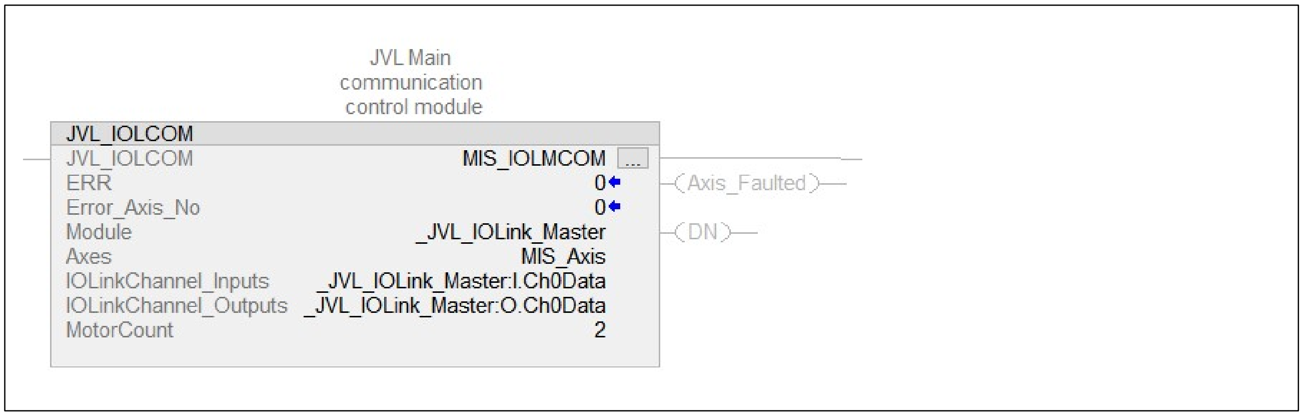
This AOI handles basic data exchange, value scaling, configuration and fault handling.

When EthenetIP is used a separate process handles the cyclic communication with the motor, using IOLInk the communication is handles by the JVL gateway, but the addressing and data payload is handled by the JVL_IOLCOM –AOI.

The AOI must be scanned continuously and can be placed in the Mainroutine.
In case of an fault the “Axis Faulted” –output will be set and the parameter “Error_Axis_No” will indicate at which motor address the problem occurred.



```
JVL Main
communication
control module
JVL_IOLCOM
JVL_IOLCOM                                   MIS_IOLMCOM  ...
ERR                                                    0      -(Axis_Faulted)-
Error_Axis_No                                          0
Module                                 _JVL_IOLink_Master      -(DN)-
Axes                                             MIS_Axis
IOLinkChannel_Inputs         _JVL_IOLink_Master:I.Ch0Data
IOLinkChannel_Outputs        _JVL_IOLink_Master:O.Ch0Data
MotorCount                                             2
```



```
JVL Main
communication
control module
JVL_IOLCOM
JVL_IOLCOM                                   MIS_IOLMCOM  ...
ERR                                                    0      -(Axis_Faulted)-
Error_Axis_No                                          0
Module                                 _JVL_IOLink_Master      -(DN)-
Axes                                             MIS_Axis
IOLinkChannel_Inputs         _JVL_IOLink_Master:I.Ch0Data
IOLinkChannel_Outputs        _JVL_IOLink_Master:O.Ch0Data
MotorCount                                             2
```

JVL_IOLCOM parameters
JVL_IOLCOM: Instance of the actual AOI
ERR: Ongoing error code
Error_Axis_No: Which motor address the fault occurred at.
Module: IOLink master module in project.
Axes: An array of type JVL_Axis_Stepper that holds the JVL Axis structure for all the motors.
IOLinkChannel_Inputs: Cyclic inputs from iolink master module.
IOLinkChannel_Outputs: Cyclic outputs to iolink master module.
MotorCount: how many motors are daisy chained on the RS422 network. Values range is 1-5.
Axis Faulted: Will indicate if a fault was detected on a motor, the Error_Axis_No will indicate which address the fault was detected.
DN: Indicates when the initial configuration of the motors after S:FS is done.

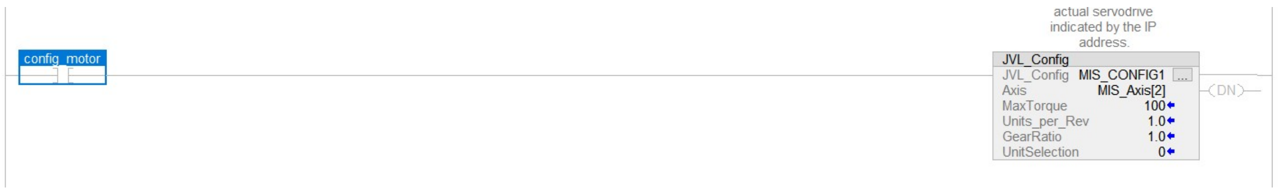
5.2 JVL_Config,

This AOI will configure the internal tags that handles the unit scaling. These tags are located in the Axis parameter.

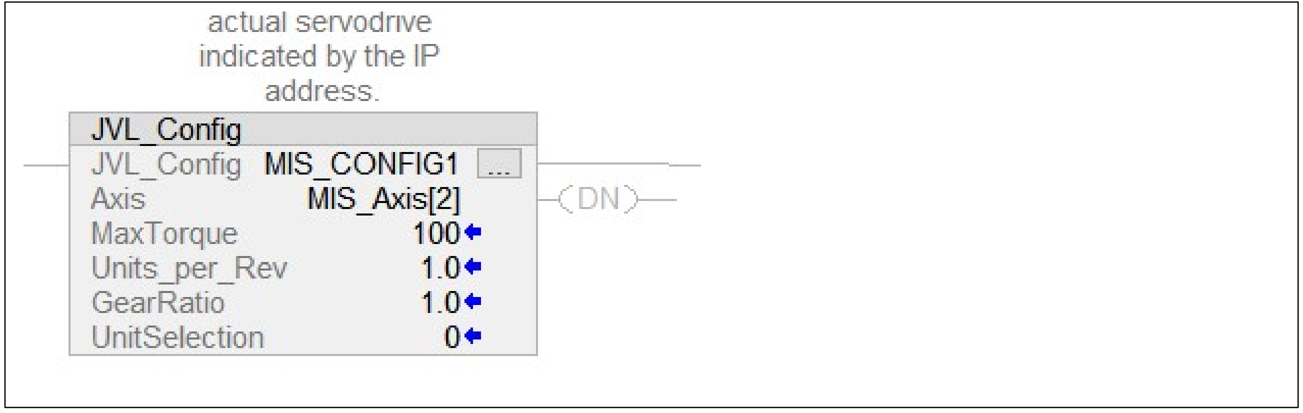
Both scaling of the actual units used and the units used are configured through this AOI. In case a gearbox is mounted on the motor shaft, the gear ratio can be entered so that the units will follow the outgoing shaft in the application.

The units will influence Velocity and position values. Acceleration and torque values are not affected by the scaling feature,

It is only necessary to call this AOI ones if the tags are saved in the PLC.



JVL_Config
JVL_Config MIS_CONFIG1
Axis MIS_Axis[2]
MaxTorque 100
Units_per_Rev 1.0
GearRatio 1.0
UnitSelection 0



JVL_Config
JVL_Config MIS_CONFIG1
Axis MIS_Axis[2]
MaxTorque 100
Units_per_Rev 1.0
GearRatio 1.0
UnitSelection 0

JVL_Config parameters
JVL_Config: Instance of the actual AOI
Axis: The actual Axis to configure, in the above screenshot Addr=2 is used.
MaxTorque: 1-100%, note that for steppers equipped with H2/H4 encoder option, closed loop and current control will control the current, so this value will be the Max. allowed torque. Motors without any encoder installed will utilize this value whenever a motion is required.
Units_per_Rev: Amount of scaled units per. Revolution. Exc. 60mm/rev. on the output shaft equals 60.0 in this parameter.
GearRatio: in case a gearbox is mounted the ratio can be entered here and the scaling will compensate the for the gearbox. Exc. A gearbox of 10:1, the value 10.0 is entered.
UnitSelection: For the units used, a number between 1 and 4 can be entered, where:

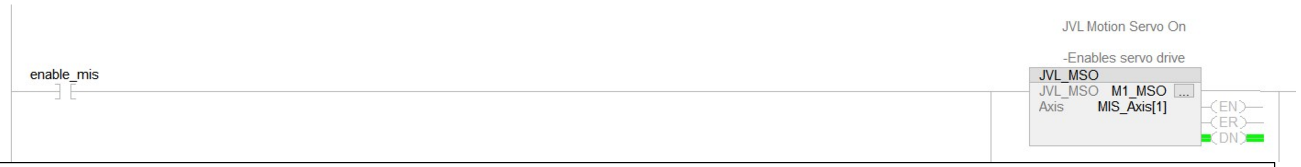
0 = None, native motor units used Velocity [1/100 Rev/min], position [Counts]
1 = Velocity in [µm/s] and Position in [µm]
2 = Velocity in [mm/s] and Position in [mm]
3 = Velocity in [m/s] and Position in [m]
4 = Velocity in [inch/s] and Position in [inch]
5 = Velocity in [Degree/s] and Position in [Degree], obs! Not turntable mode values will not wrap.

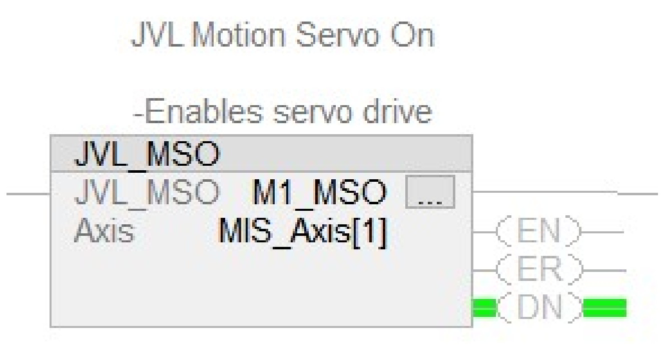
5.3 JVL_MSO, Motion Servo On

The MSO AOI basically enables the motor. All other motion AOI's will fail if the motor isn't enabled prior of calling a motion AOI.

When the MSO is called, the motor will enter Position mode and maintain the position using up to the Max. allowed torque configured in the JVL_Config AOI.

The instruction is edge triggered and DN will go high when the instruction is successfully executed.



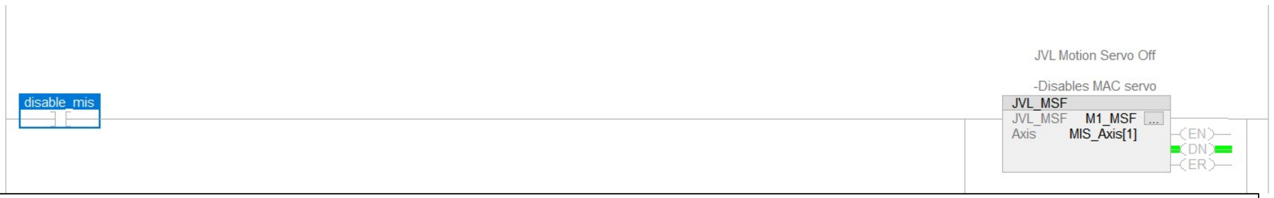


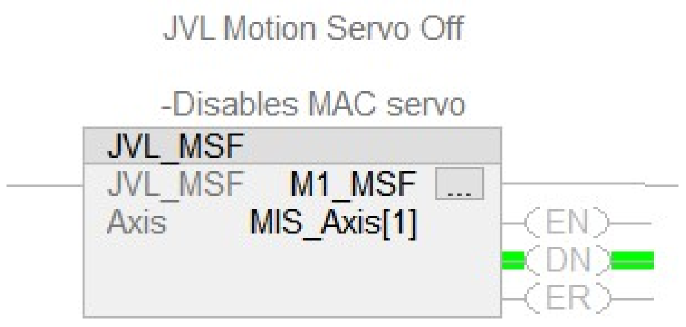
JVL_MSO parameters
JVL_MSO: Instance of the actual AOI
Axis: The actual Axis to configure, in the above screenshot Addr=1 is used.
DN: Goes high when the instruction has executed.
ER: In case the motor has already been enabled, ER will go high.
EN: Indicates that the instruction is executing.

5.4 JVL_MSF, Motion Servo Off

The JVL_MSF disables a motor and will leave the motor torque less.

The instruction will fail if the motor is disabled upon calling the instruction.

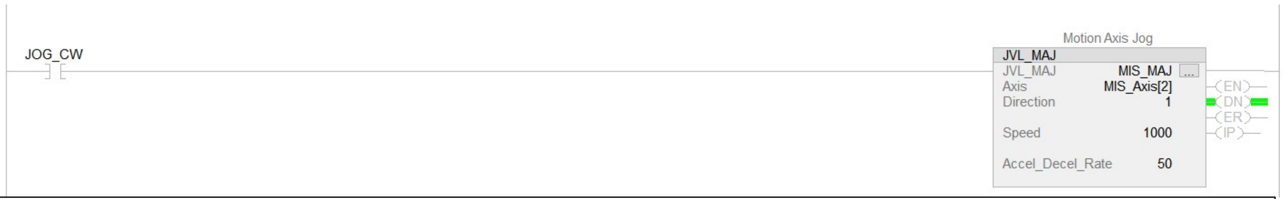


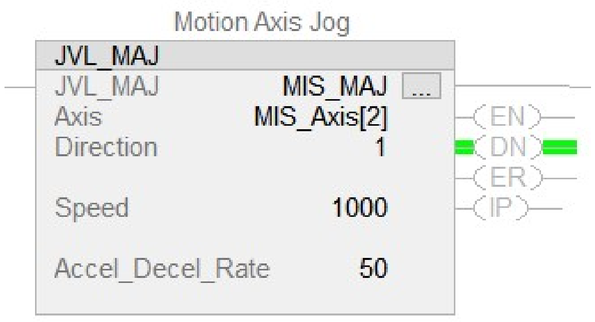


JVL_MSF parameters
JVL_MSF: Instance of the actual AOI
Axis: The actual Axis to configure, in the above screenshot Addr=1 is used.
DN: Goes high when the instruction has executed.
ER: In case the motor has already been disabled, ER will go high.
EN: Indicates that the instruction is executing.

5.5 JVL_MAJ, Motion Axis Jog

Used for Jogging purpose, after the AOI has been called the motor will continuously move until stopped by a call to the JVL_MAS –command.
The direction and velocity is configured in the parameters.



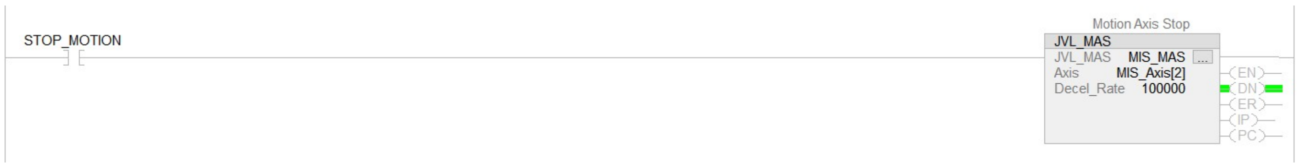


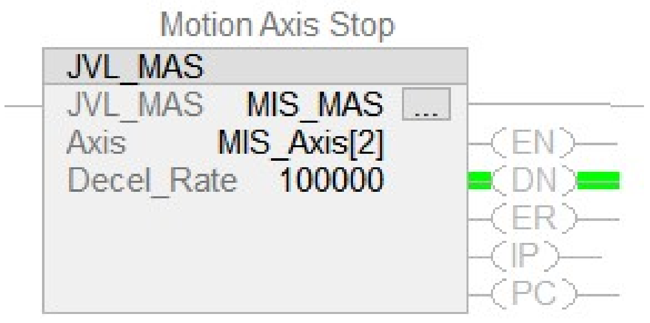
JVL_MAJ parameters
JVL_MAJ: Instance of the actual AOI
Axis: The actual Axis to configure, in the above screenshot Addr=2 is used.
Direction: 0 = CW rotation, 1 = CCW rotation.
Speed: 0 - Max. RPM in the units selected from the JVL_Config call.
Accel_Decel_Rate: Acceleration and deceleration rate 50.0 – 2^31
EN: Goes high when activated.
DN: The instruction has been executed
ER: An error occurred, the ERR of the JVL_MAJ instance will hold an additional errorcode.
IP: Motion is In Progress.

ERR errorcodes:
1: Direction value out of range [0..1]
2: Speed out of range [0..Max.]
4: Acceleration/Deceleration value out of range, (must be > 0)
5: Motor is disabled, call JVL_MSO

5.6 JVL_MAS, Motion Axis Stop

When a motion is initiated, the motion can be stopped by calling the JVL_MAS instruction.



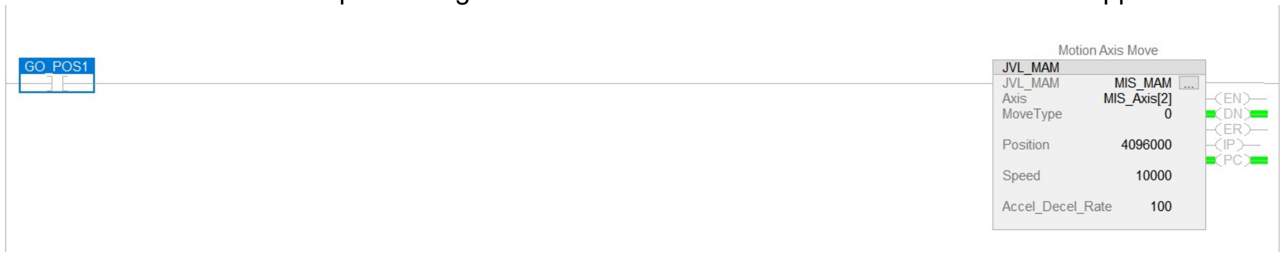


JVL_MAS parameters
JVL_MAS: Instance of the actual AOI
Axis: The actual Axis to configure, in the above screenshot Addr=2 is used.
Accel_Decel_Rate: Acceleration and deceleration rate 50.0 – 2^31
EN: Goes high when activated.
DN: The instruction has been executed
ER: An error occurred, the ERR of the JVL_MAJ instance will hold an additional errorcode.
IP: Motion is In Progress.
PC: Process Complete, Motion has stopped.

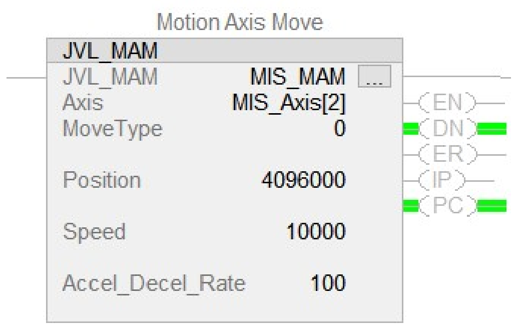
ERR errorcodes:
2: Acceleration/Deceleration value out of range, (must be > 0)
5: Motor is disabled

5.7 JVL_MAM, Motion Axis Move

The MAM instruction handles positioning of the shaft. Both absolute and relative motion is supported.



JVL_MAM: MIS_MAM
Axis: MIS_Axis[2]
MoveType: 0
Position: 4096000
Speed: 10000
Accel_Decel_Rate: 100



JVL_MAM: MIS_MAM
Axis: MIS_Axis[2]
MoveType: 0
Position: 4096000
Speed: 10000
Accel_Decel_Rate: 100

JVL_MAM parameters
JVL_MAM: Instance of the actual AOI
Axis: The actual Axis to configure, in the above screenshot Addr=2 is used.
MoveType: 0=Absolute, 1=Relative
Position: The scaled or non scaled position.
Speed: Scaled or non scaled speed.
Accel_Decel_Rate: Acceleration and deceleration rate, (must be > 0)

EN: Goes high when activated.
DN: The instruction has been executed
ER: An error occurred, the ERR will hold an additional errorcode.
IP: Motion is In Progress.
PC: Process Complete, Motion has stopped.

ERR –codes:
5: Motor is disabled.
13: Parameter out of range, see EXERR –codes for details.

EXERR codes:
2: MoveType is out of range, must 0 or 1.
3: Position is out of range, value must be a signed 32bit value.
4: Speed is out of range, value must be > 0.
5: Accel_Decel_Rate out of range, value must be > 0

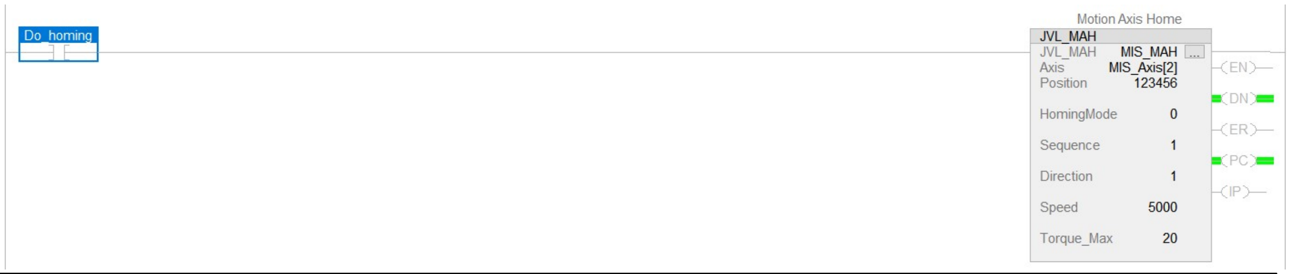
5.8 JVL_MAH, Motion Axis Homing

The MAH instruction is used, when the motor position needs to be referenced. Passive or active homing is supported, where passive homing presets the shaft position to a certain value, without moving the shaft. Active homing moves the shaft until an event of either a sensor input (in the motor) is triggered or a torque threshold is exceeded.

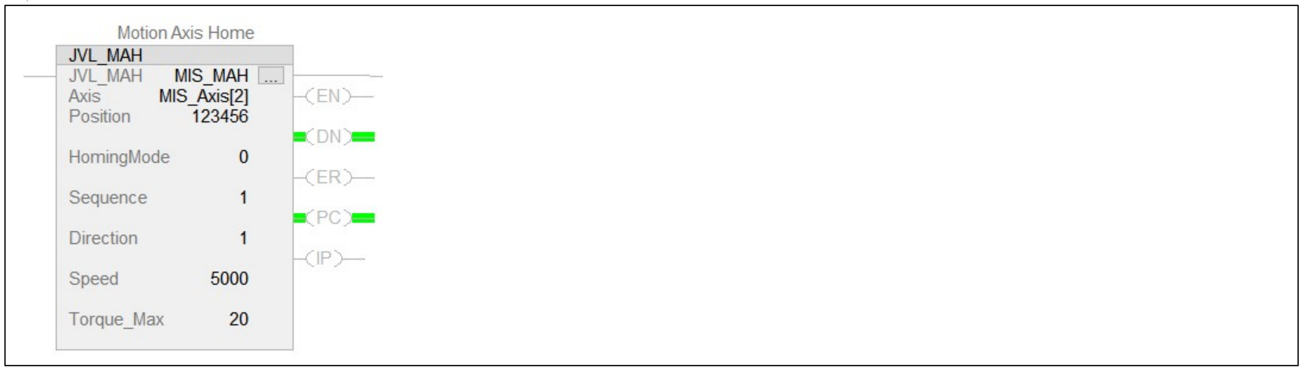
The homing methods are defined by 2 parameters, Sequence and Direction.

Supported homing methods:

Sequence	Direction	Description
1	0	Forward homing to switch, Uni direction. Stops on switch rising edge.
1	1	Forward homing to switch, Bi direction. Detects switch on rising edge and returns to find falling edge. This is a more accurate method than the Uni direction – method.
1	2	Reverse homing to switch, Uni direction. Stops on switch rising edge.
1	3	Reverse homing to switch, Bi direction. Detects switch on rising edge and returns to find falling edge. This is a more accurate method than the Uni direction – method.
4	0	Forward Torque homing. Trips when the actual torque > Torque_Max [%]
4	2	Reverse Torque homing. Trips when the actual torque > Torque_Max [%]



Motion Axis Home
JVL_MAH
JVL_MAH MIS_MAH
Axis MIS_Axis[2]
Position 123456
HomingMode 0
Sequence 1
Direction 1
Speed 5000
Torque_Max 20



Motion Axis Home
JVL_MAH
JVL_MAH MIS_MAH
Axis MIS_Axis[2]
Position 123456
HomingMode 0
Sequence 1
Direction 1
Speed 5000
Torque_Max 20

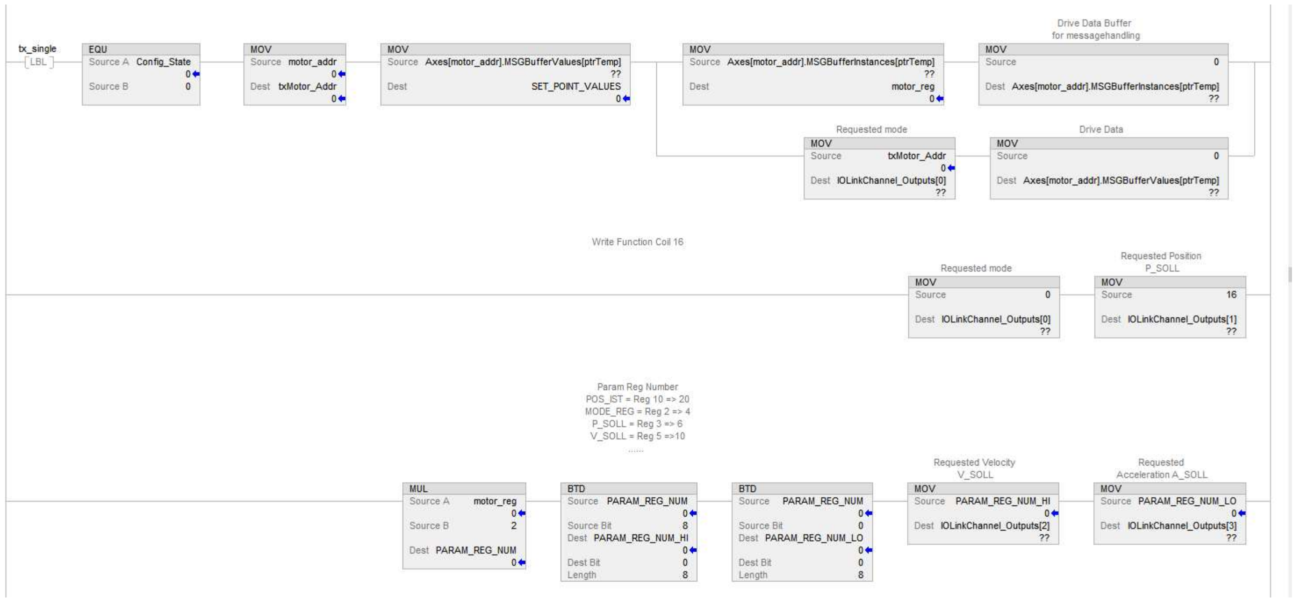
Continued…

JVL_MAH parameters
JVL_MAH: Instance of the actual AOI
Axis: The actual Axis to configure, in the above screenshot Addr=2 is used.
Position: The position reference at the homing point.
HomingMode: 0=Passive, 1=Active
Sequence: Selects the method used for the homing process along with the Direction parameter. See the table for further details on which homing method to use and how to setup the Sequence and Direction parameters.
Speed: With the Sequence parameter, Direction selects the homing method used.
Position: The scaled or non scaled position.
Speed: Scaled or non scaled homing speed.
Torque_Max: Trip point at which Torque homing is triggered.

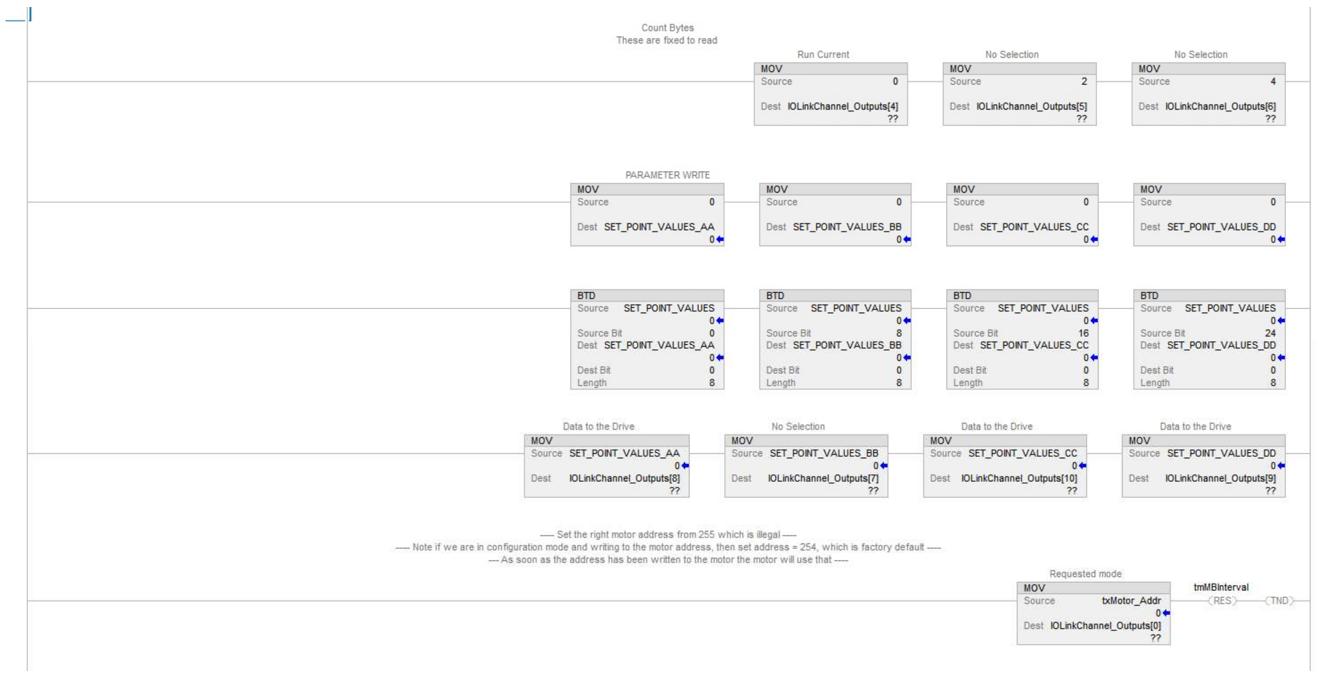
EN: Goes high when activated.
DN: The instruction has been executed
ER: An error occurred, the ERR will hold an additional errorcode.
IP: Motion is In Progress.
PC: Process Complete, Motion has stopped.

ERR –codes:
5: Motor is disabled.
13: Parameter out of range, see EXERR –codes for details.

EXERR codes:
2: HomeMode out of range, Value must be 0=Passive, 1=Active.
3: Position is out of range, value must be a signed 32bit value.
4: Offset is out of range, value must be a signed 32bit value.
5: Sequence is out of range, must be 1 or 4
7: Direction is out of range for Sequence = 1,Value must be within 0-3
8: Speed out of range, value must be > 0
10: Torque_Max out of range, value must be within 0-90%

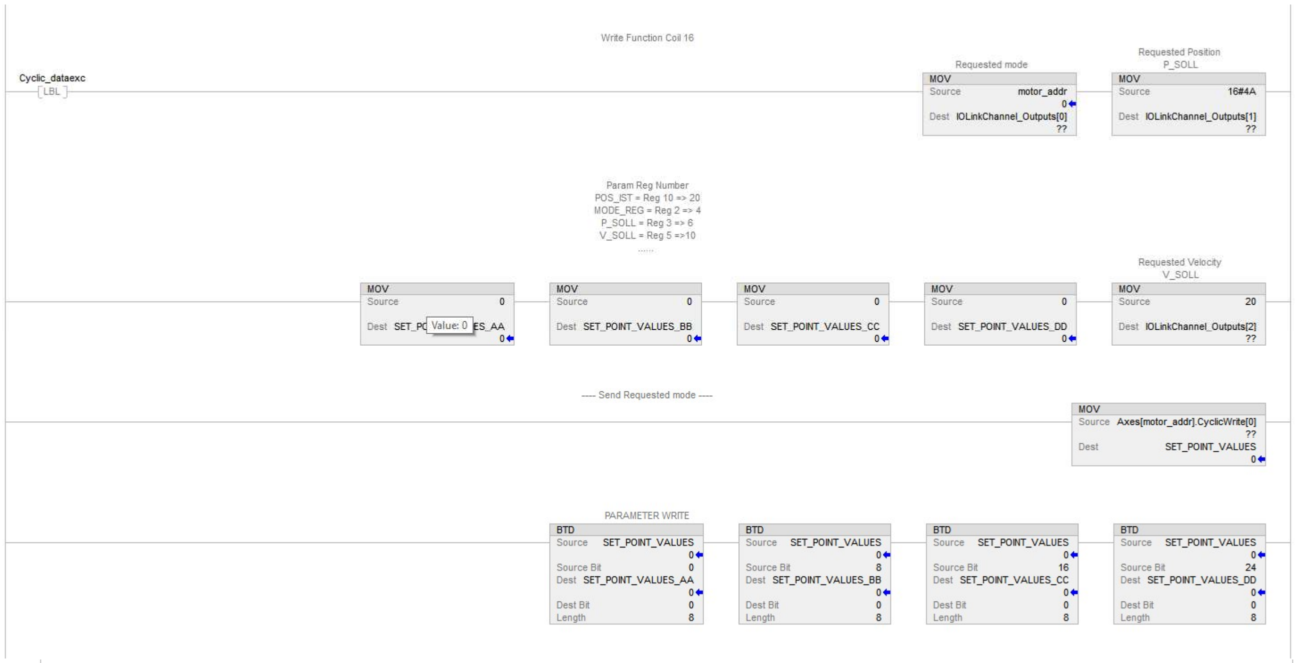


```
tx_single
-[LBL]-
EQU   Source A Config_State 0   Source B 0
MOV   Source motor_addr 0   Dest txMotor_Addr 0
MOV   Source Axes[motor_addr].MSGBufferValues[ptrTemp] ??   Dest SET_POINT_VALUES 0
MOV   Source Axes[motor_addr].MSGBufferInstances[ptrTemp] ??   Dest motor_reg 0
Drive Data Buffer for messagehandling
MOV   Source 0   Dest Axes[motor_addr].MSGBufferInstances[ptrTemp] ??
Requested mode
MOV   Source txMotor_Addr 0   Dest IOLinkChannel_Outputs[0] ??
Drive Data
MOV   Source 0   Dest Axes[motor_addr].MSGBufferValues[ptrTemp] ??

Write Function Coil 16
Requested mode
MOV   Source 0   Dest IOLinkChannel_Outputs[0] ??
Requested Position P_SOLL
MOV   Source 16   Dest IOLinkChannel_Outputs[1] ??

Param Reg Number
POS_IST = Reg 10 => 20
MODE_REG = Reg 2 => 4
P_SOLL = Reg 3 => 6
V_SOLL = Reg 5 =>10
......
MUL   Source A motor_reg 0   Source B 2   Dest PARAM_REG_NUM 0
BTD   Source PARAM_REG_NUM 0   Source Bit 8   Dest PARAM_REG_NUM_HI 0   Dest Bit 0   Length 8
BTD   Source PARAM_REG_NUM 0   Source Bit 0   Dest PARAM_REG_NUM_LO 0   Dest Bit 0   Length 8
Requested Velocity V_SOLL
MOV   Source PARAM_REG_NUM_HI 0   Dest IOLinkChannel_Outputs[2] ??
Requested Acceleration A_SOLL
MOV   Source PARAM_REG_NUM_LO 0   Dest IOLinkChannel_Outputs[3] ??
```

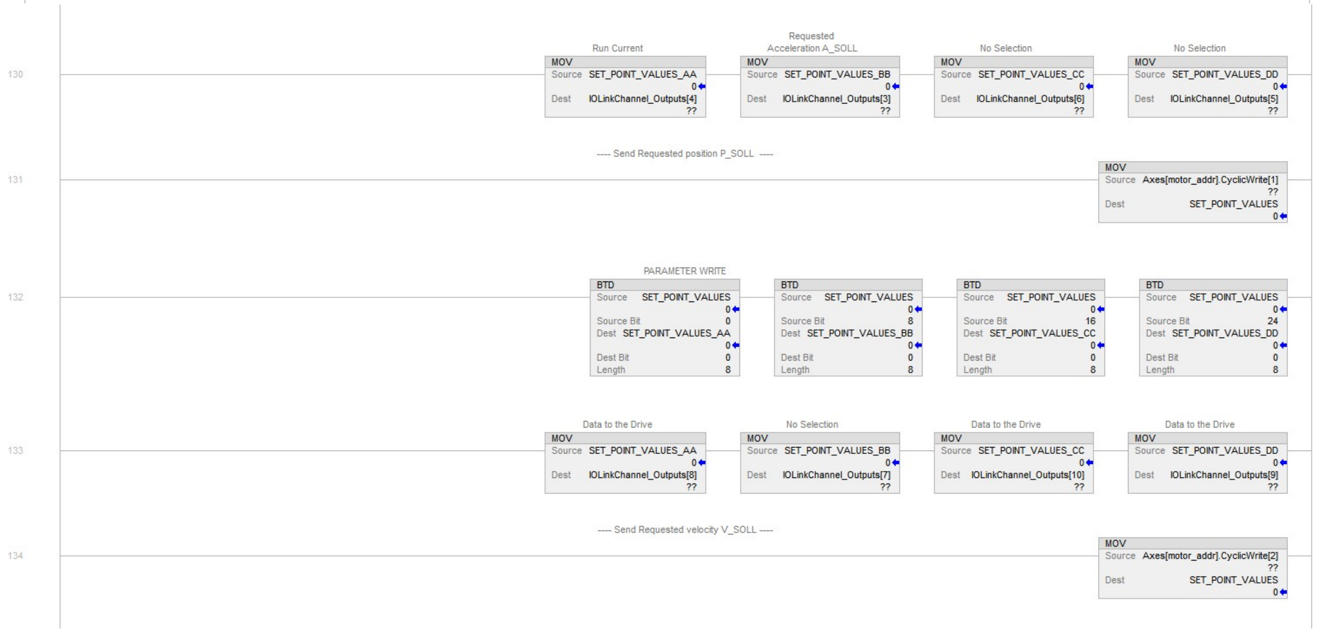


```
Count Bytes
These are fixed to read
Run Current
MOV   Source 0   Dest IOLinkChannel_Outputs[4] ??
No Selection
MOV   Source 2   Dest IOLinkChannel_Outputs[5] ??
No Selection
MOV   Source 4   Dest IOLinkChannel_Outputs[6] ??

PARAMETER WRITE
MOV   Source 0   Dest SET_POINT_VALUES_AA 0
MOV   Source 0   Dest SET_POINT_VALUES_BB 0
MOV   Source 0   Dest SET_POINT_VALUES_CC 0
MOV   Source 0   Dest SET_POINT_VALUES_DD 0

BTD   Source SET_POINT_VALUES 0   Source Bit 0   Dest SET_POINT_VALUES_AA 0   Dest Bit 0   Length 8
BTD   Source SET_POINT_VALUES 0   Source Bit 8   Dest SET_POINT_VALUES_BB 0   Dest Bit 0   Length 8
BTD   Source SET_POINT_VALUES 0   Source Bit 16   Dest SET_POINT_VALUES_CC 0   Dest Bit 0   Length 8
BTD   Source SET_POINT_VALUES 0   Source Bit 24   Dest SET_POINT_VALUES_DD 0   Dest Bit 0   Length 8

Data to the Drive
MOV   Source SET_POINT_VALUES_AA 0   Dest IOLinkChannel_Outputs[8] ??
No Selection
MOV   Source SET_POINT_VALUES_BB 0   Dest IOLinkChannel_Outputs[7] ??
Data to the Drive
MOV   Source SET_POINT_VALUES_CC 0   Dest IOLinkChannel_Outputs[10] ??
Data to the Drive
MOV   Source SET_POINT_VALUES_DD 0   Dest IOLinkChannel_Outputs[9] ??

---- Set the right motor address from 255 which is illegal ----
---- Note if we are in configuration mode and writing to the motor address, then set address = 254, which is factory default ----
---- As soon as the address has been written to the motor the motor will use that ----
Requested mode
MOV   Source txMotor_Addr 0   Dest IOLinkChannel_Outputs[0] ??
tmMBInterval
-(RES)-   -(TND)-
```

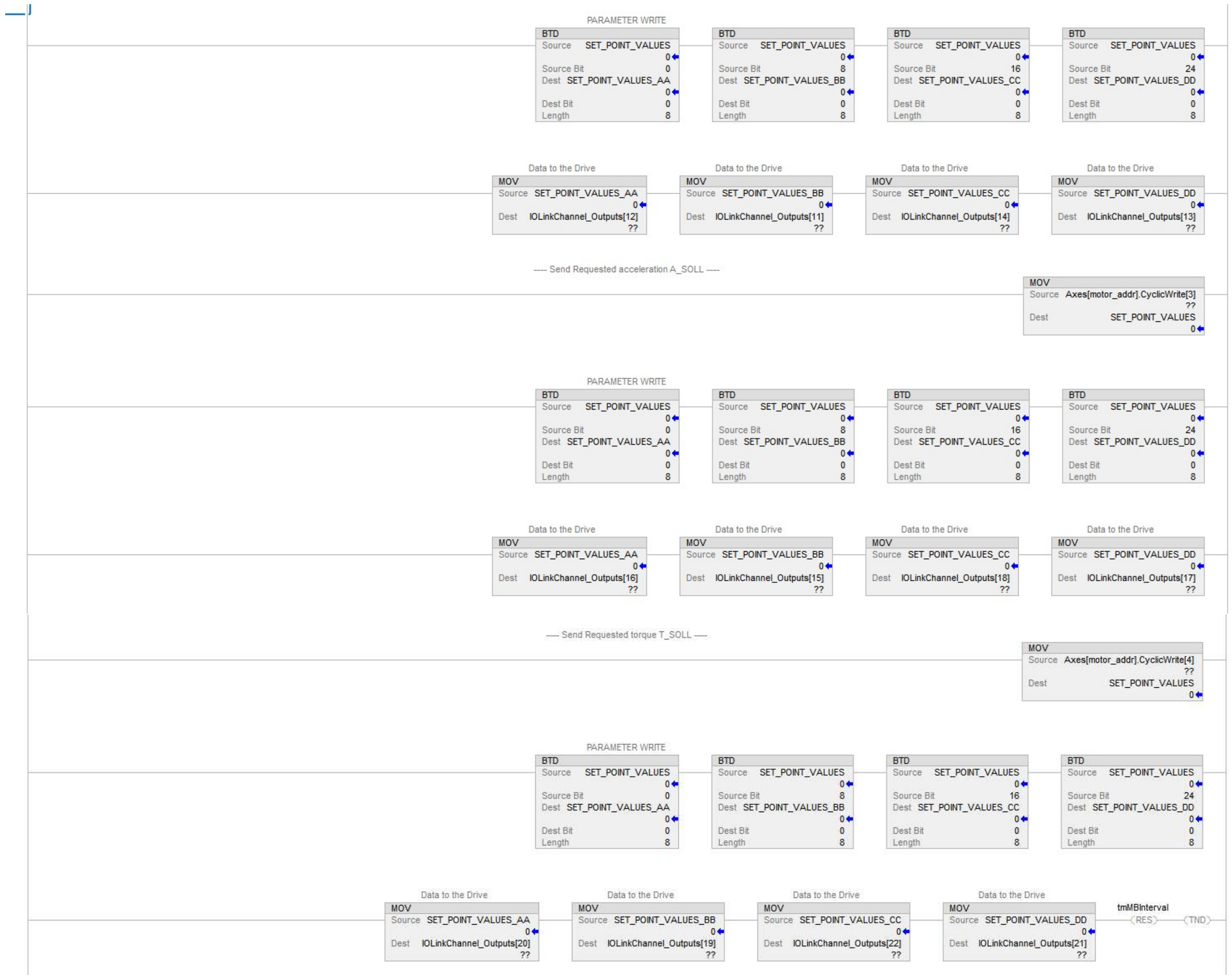
PDO



```
Write Function Coil 16

Cyclic_dataexc
[LBL]

Requested mode
MOV
Source        motor_addr
                       0
Dest  IOLinkChannel_Outputs[0]
                      ??

Requested Position
P_SOLL
MOV
Source              16#4A
Dest  IOLinkChannel_Outputs[1]
                      ??

Param Reg Number
POS_IST = Reg 10 => 20
MODE_REG = Reg 2 => 4
P_SOLL = Reg 3 => 6
V_SOLL = Reg 5 =>10
......

MOV
Source                 0
Dest  SET_PC Value: 0 ES_AA
                       0

MOV
Source                 0
Dest  SET_POINT_VALUES_BB
                       0

MOV
Source                 0
Dest  SET_POINT_VALUES_CC
                       0

MOV
Source                 0
Dest  SET_POINT_VALUES_DD
                       0

Requested Velocity
V_SOLL
MOV
Source                20
Dest  IOLinkChannel_Outputs[2]
                      ??

---- Send Requested mode ----
MOV
Source  Axes[motor_addr].CyclicWrite[0]
                      ??
Dest          SET_POINT_VALUES
                       0

PARAMETER WRITE
BTD
Source    SET_POINT_VALUES
                       0
Source Bit             0
Dest  SET_POINT_VALUES_AA
                       0
Dest Bit               0
Length                 8

BTD
Source    SET_POINT_VALUES
                       0
Source Bit             8
Dest  SET_POINT_VALUES_BB
                       0
Dest Bit               0
Length                 8

BTD
Source    SET_POINT_VALUES
                       0
Source Bit            16
Dest  SET_POINT_VALUES_CC
                       0
Dest Bit               0
Length                 8

BTD
Source    SET_POINT_VALUES
                       0
Source Bit            24
Dest  SET_POINT_VALUES_DD
                       0
Dest Bit               0
Length                 8
```



```
Run Current
MOV
Source  SET_POINT_VALUES_AA
                       0
Dest  IOLinkChannel_Outputs[4]
                      ??

Requested
Acceleration A_SOLL
MOV
Source  SET_POINT_VALUES_BB
                       0
Dest  IOLinkChannel_Outputs[3]
                      ??

No Selection
MOV
Source  SET_POINT_VALUES_CC
                       0
Dest  IOLinkChannel_Outputs[6]
                      ??

No Selection
MOV
Source  SET_POINT_VALUES_DD
                       0
Dest  IOLinkChannel_Outputs[5]
                      ??

---- Send Requested position P_SOLL ----
MOV
Source  Axes[motor_addr].CyclicWrite[1]
                      ??
Dest          SET_POINT_VALUES
                       0

PARAMETER WRITE
BTD
Source    SET_POINT_VALUES
                       0
Source Bit             0
Dest  SET_POINT_VALUES_AA
                       0
Dest Bit               0
Length                 8

BTD
Source    SET_POINT_VALUES
                       0
Source Bit             8
Dest  SET_POINT_VALUES_BB
                       0
Dest Bit               0
Length                 8

BTD
Source    SET_POINT_VALUES
                       0
Source Bit            16
Dest  SET_POINT_VALUES_CC
                       0
Dest Bit               0
Length                 8

BTD
Source    SET_POINT_VALUES
                       0
Source Bit            24
Dest  SET_POINT_VALUES_DD
                       0
Dest Bit               0
Length                 8

Data to the Drive
MOV
Source  SET_POINT_VALUES_AA
                       0
Dest  IOLinkChannel_Outputs[8]
                      ??

No Selection
MOV
Source  SET_POINT_VALUES_BB
                       0
Dest  IOLinkChannel_Outputs[7]
                      ??

Data to the Drive
MOV
Source  SET_POINT_VALUES_CC
                       0
Dest  IOLinkChannel_Outputs[10]
                      ??

Data to the Drive
MOV
Source  SET_POINT_VALUES_DD
                       0
Dest  IOLinkChannel_Outputs[9]
                      ??

---- Send Requested velocity V_SOLL ----
MOV
Source  Axes[motor_addr].CyclicWrite[2]
                      ??
Dest          SET_POINT_VALUES
                       0
```


```
PARAMETER WRITE
BTD                              BTD                              BTD                              BTD
Source  SET_POINT_VALUES         Source  SET_POINT_VALUES         Source  SET_POINT_VALUES         Source  SET_POINT_VALUES
                       0                                0                                0                                0
Source Bit             0         Source Bit             8         Source Bit            16         Source Bit            24
Dest  SET_POINT_VALUES_AA        Dest  SET_POINT_VALUES_BB        Dest  SET_POINT_VALUES_CC        Dest  SET_POINT_VALUES_DD
                       0                                0                                0                                0
Dest Bit               0         Dest Bit               0         Dest Bit               0         Dest Bit               0
Length                 8         Length                 8         Length                 8         Length                 8

Data to the Drive                    Data to the Drive                    Data to the Drive                    Data to the Drive
MOV                                  MOV                                  MOV                                  MOV
Source  SET_POINT_VALUES_AA          Source  SET_POINT_VALUES_BB          Source  SET_POINT_VALUES_CC          Source  SET_POINT_VALUES_DD
                          0                                    0                                    0                                    0
Dest  IOLinkChannel_Outputs[12]      Dest  IOLinkChannel_Outputs[11]      Dest  IOLinkChannel_Outputs[14]      Dest  IOLinkChannel_Outputs[13]
                         ??                                   ??                                   ??                                   ??

---- Send Requested acceleration A_SOLL ----
MOV
Source  Axes[motor_addr].CyclicWrite[3]
                                     ??
Dest                   SET_POINT_VALUES
                                      0

PARAMETER WRITE
BTD                              BTD                              BTD                              BTD
Source  SET_POINT_VALUES         Source  SET_POINT_VALUES         Source  SET_POINT_VALUES         Source  SET_POINT_VALUES
                       0                                0                                0                                0
Source Bit             0         Source Bit             8         Source Bit            16         Source Bit            24
Dest  SET_POINT_VALUES_AA        Dest  SET_POINT_VALUES_BB        Dest  SET_POINT_VALUES_CC        Dest  SET_POINT_VALUES_DD
                       0                                0                                0                                0
Dest Bit               0         Dest Bit               0         Dest Bit               0         Dest Bit               0
Length                 8         Length                 8         Length                 8         Length                 8

Data to the Drive                    Data to the Drive                    Data to the Drive                    Data to the Drive
MOV                                  MOV                                  MOV                                  MOV
Source  SET_POINT_VALUES_AA          Source  SET_POINT_VALUES_BB          Source  SET_POINT_VALUES_CC          Source  SET_POINT_VALUES_DD
                          0                                    0                                    0                                    0
Dest  IOLinkChannel_Outputs[16]      Dest  IOLinkChannel_Outputs[15]      Dest  IOLinkChannel_Outputs[18]      Dest  IOLinkChannel_Outputs[17]
                         ??                                   ??                                   ??                                   ??

---- Send Requested torque T_SOLL ----
MOV
Source  Axes[motor_addr].CyclicWrite[4]
                                     ??
Dest                   SET_POINT_VALUES
                                      0

PARAMETER WRITE
BTD                              BTD                              BTD                              BTD
Source  SET_POINT_VALUES         Source  SET_POINT_VALUES         Source  SET_POINT_VALUES         Source  SET_POINT_VALUES
                       0                                0                                0                                0
Source Bit             0         Source Bit             8         Source Bit            16         Source Bit            24
Dest  SET_POINT_VALUES_AA        Dest  SET_POINT_VALUES_BB        Dest  SET_POINT_VALUES_CC        Dest  SET_POINT_VALUES_DD
                       0                                0                                0                                0
Dest Bit               0         Dest Bit               0         Dest Bit               0         Dest Bit               0
Length                 8         Length                 8         Length                 8         Length                 8

Data to the Drive                    Data to the Drive                    Data to the Drive                    Data to the Drive
MOV                                  MOV                                  MOV                                  MOV
Source  SET_POINT_VALUES_AA          Source  SET_POINT_VALUES_BB          Source  SET_POINT_VALUES_CC          Source  SET_POINT_VALUES_DD
                          0                                    0                                    0                                    0
Dest  IOLinkChannel_Outputs[20]      Dest  IOLinkChannel_Outputs[19]      Dest  IOLinkChannel_Outputs[22]      Dest  IOLinkChannel_Outputs[21]
                         ??                                   ??                                   ??                                   ??
                                                                                                         tmMBInterval
                                                                                                         -(RES)-        -(TND)-
```