Pacdrive3 - Lxm62-Servoantrieb

1. SPS-Konfiguration

Die Achse muss als Lexium62-Linearantrieb im Machine Expert eingestellt werden, und die Motoridentifikation muss als Motor ohne Typenschild eingestellt werden.

2. Einstellen des Typenschilds

```
st_MotorDataRW_01.i_xEnable := TRUE;                        // Enable the write function
st_MotorDataRW_01.i_etStorageLocation := mtp.ET_StorageLocation.Drive;  //  Storage Location (encoder or drive)
st_MotorDataRW_01.i_sFilename := 'ide0:MDF/UserMotorData.mdf';      //  file name of the typlete for drive
st_MotorDataRW_01.i_sFilenameBLH := 'ide0:MDF/';            //      file name of the typlete for encoder
st_UserMotorData_01.etMotorType := MTP.ET_MotorType.LinearPMSM;     // type of the motor
st_UserMotorData_01.sMotorname := 'Nytek';  //Armonic Drive         //  string of the motor name
st_UserMotorData_01.sMotorSerialNumber := '33333';              //  string of serial number
st_UserMotorData_01.sMotorArticleNumber := '444444';            //      string of article number
st_UserMotorData_01.stMotorDataPMSM.uiEncoderType := mtp.ET_EncoderType.SincosLinear; //    encoder type
st_UserMotorData_01.stMotorDataPMSM.uiNominalSpeed := 3998; // mm/s
st_UserMotorData_01.stMotorDataPMSM.rNominalVoltage := 220.0;
st_UserMotorData_01.stMotorDataPMSM.rNominalCurrent := 2.1; // A
st_UserMotorData_01.stMotorDataPMSM.rPeakCurrent := 8.0; // A , OPTIONAL, std: Nom*1.5
st_UserMotorData_01.stMotorDataPMSM.rContStallCurrent := 2.1; // A
st_UserMotorData_01.stMotorDataPMSM.rConstStallTorque := 110; // N, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.rPeakTorque := 880; // Nm, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.rPhaseResistance := 12; // Ohm , OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.rQuadraturePhaseInductance := 18000; // uH , OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.rDirectPhaseInductance := 16200; // uH
st_UserMotorData_01.stMotorDataPMSM.uiRotatingFieldDirection := 0; // OPTIONAL, std: 0
st_UserMotorData_01.stMotorDataPMSM.rEMK_Constant := 18;


(*************************************************)
st_UserMotorData_01.stMotorDataPMSM.uiPolePair := 1;                Number of pole pairs
(*************************************************)


st_UserMotorData_01.stMotorDataPMSM.uiMaxMotorTemperature := 90; //°C , OPTIONAL, std: 130
st_UserMotorData_01.stMotorDataPMSM.uiTempSensorType := 1;
st_UserMotorData_01.stMotorDataPMSM.uiTempSensorResistanceOvertemp := 4000; // Ohm, OPTIONAL (verw bei SensorType = 1), std: 4000
st_UserMotorData_01.stMotorDataPMSM.auiTempSensorCharacteristic[0] := 0; // Ohm, OPTIONAL (verw bei SensorType = 2)
st_UserMotorData_01.stMotorDataPMSM.rInsulationSystemVoltage := 0; // V, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiEncoderMaxSpeed := 0; // Umin, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiEncoderMaxTemp := 0; // °C, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiEncoderTempSensor := 0; // OPTIONAL, std: 0
st_UserMotorData_01.stMotorDataPMSM.uiEncoderNumberOfTurns :=0; // Obligatorisch für Geber ohne Hiperface
st_UserMotorData_01.stMotorDataPMSM.uiEncoderLinesPerRevolution := 0; // Obligatorisch für Geber ohne Hiperface
st_UserMotorData_01.stMotorDataPMSM.uiModelC1 := 0; // OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiModelA12 := 0; // OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiModelDeltaA := 0; // OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiModelDeltaB := 0; // OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiMaxSpeed := 2500; // mm/s
//st_UserMotorData_01.stMotorDataPMSM.udiMotorIntertia := 6300; // g
st_UserMotorData_01.stMotorDataPMSM.udiMotorInertia := 1500;
st_UserMotorData_01.stMotorDataPMSM.uiBrake := 0;
st_UserMotorData_01.stMotorDataPMSM.uiBrakeDisconnectionTime := 0;
st_UserMotorData_01.stMotorDataPMSM.uiBrakeCouplingTime := 0; // ms, OPTIONAL, std: 100
st_UserMotorData_01.stMotorDataPMSM.rBrakeMinVoltage := 0; // V, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.rBrakeMaxVoltage := 0; // V, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiBrakeNomCurrent := 0; // V, OPTIONAL
st_UserMotorData_01.stMotorDataPMSM.uiThermalConstant := 1000; // ms, OPTIONAL, std: 1000

// For Linear Motors

st_UserMotorData_01.stMotorDataPMSM.rPeriodLength := 30000.0;           Equal to polepairpitch
st_UserMotorData_01.stMotorDataPMSM.rPolePairPitch := 30000.0;          length of N-S poles
```

3. Typenschild-Datei in die Flash-Karte schreiben

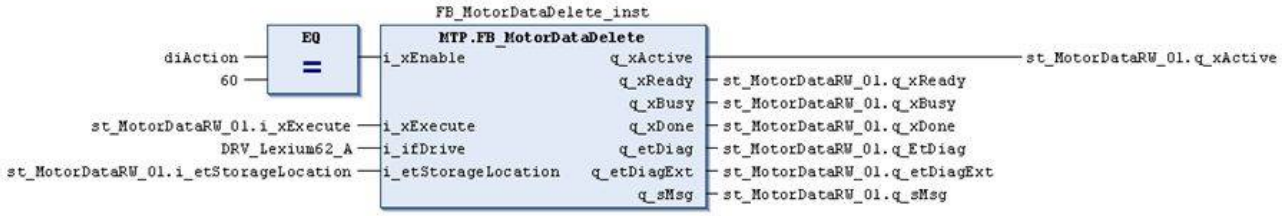
Um die Typenschild-Datei in den Flash zu erstellen, müssen Sie diese Funktion verwenden:

```
xRet := MTP.FC_MotorDataFileCreate(i_sFilename := st_MotorDataRW_01.i_sFilename, i_stUserMotorData := st_UserMotorData_01,
                                   q_sMsg => sMsg_t,
                                   q_etDiag => etDiag_t);
```

Sie erstellt eine Datei im Flash mit den Daten der Struktur.

4. Vorhandene Daten im Antrieb löschen

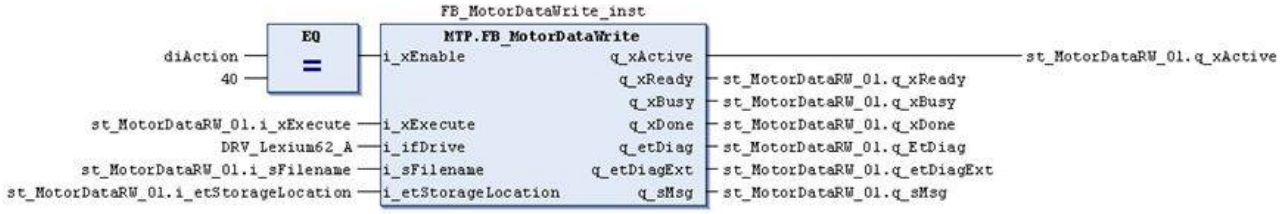
Vor allem müssen Sie die Sercos-Phase auf 2 setzen.
Dann müssen Sie alle Daten im Antrieb mit diesem Funktionsblock löschen.



Wenn keine Daten vorhanden sind, zeigt das Ergebnis diese Situation. Wenn der gesamte Prozess korrekt läuft, lautet die Meldung „Done“.

5. Typenschild in den Antrieb schreiben

Sercos weiterhin in Phase 2.
Zum Schreiben der Daten müssen Sie den Funktionsblock verwenden.



6. Antrieb ausschalten

Sercos-Phase auf Phase 0 und dann auf Phase 4 setzen. Oder Sie schalten den Antrieb AUS - EIN.
Wenn alles in Ordnung ist, erhalten Sie grünes Licht am Antrieb.

7. Motor-Kommutierung

Stromversorgung einschalten.
PowerSupplyCheckSet des PSD auf TRUE setzen. Auf 2 setzen / MotorCommutationControl testen.

Der Motor sollte sich bewegen und das Kommutierungsverfahren starten.
Am Ende liefert Ihnen MotorCommutationState das richtige Feedback.

MotorCommutationControl auf 0 und PowerSupplyCheckSet auf AUS setzen.