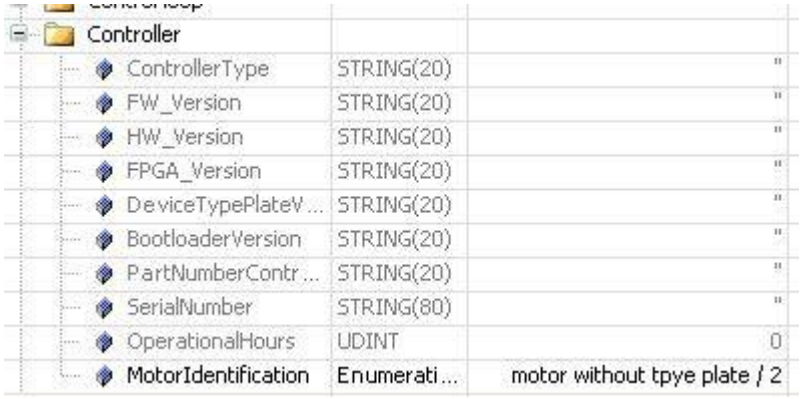
Pacdrive3 - Lxm62 servo drive

1. PLC Configuration

The axis must be set as Lexium62 linear drive inside the Machine Expert and the motor indentification must be set as motor without type plate.



Controller		
ControllerType	STRING(20)	''
FW_Version	STRING(20)	''
HW_Version	STRING(20)	''
FPGA_Version	STRING(20)	''
DeviceTypePlateV...	STRING(20)	''
BootloaderVersion	STRING(20)	''
PartNumberContr...	STRING(20)	''
SerialNumber	STRING(80)	''
OperationalHours	UDINT	0
MotorIdentification	Enumerati...	motor without tpye plate / 2

2. Setting the typeplate

```
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. Write type plate file inside the flash card

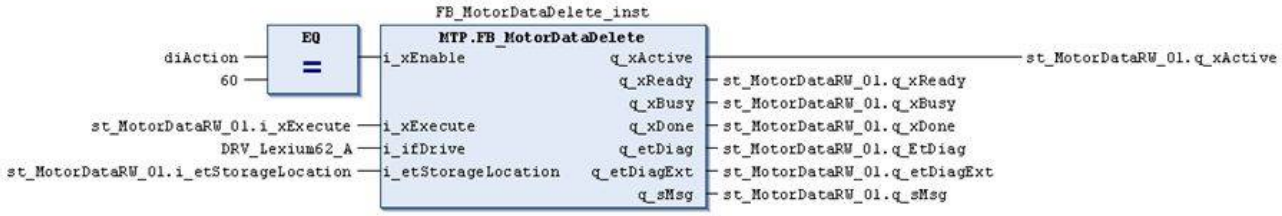
To create the typeplate file inside the flash you must use this function:

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

It create a file inside the flash with the data of the structure

4. Delete actual data inside the drive

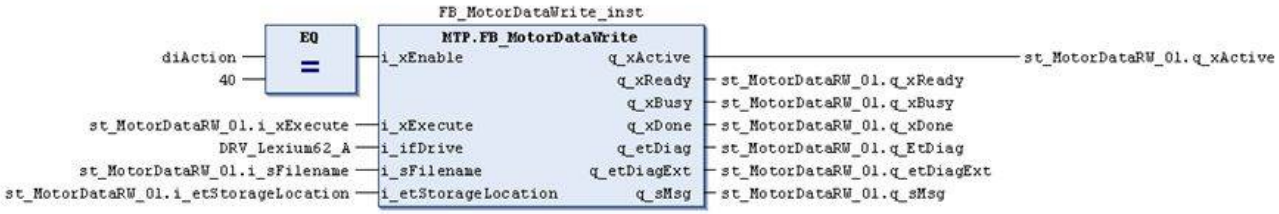
Before all you must set the sercos phase to 2
Then you must delete any data inside the drive with this function block.



If there is no data inside the result show this situation. If all the process goes right the msg is “Done”

5. Write typeplate inside the drive

Sercos still in phase 2
To write the data you must use the function block.



6. Turn off the drive

Set Sercos Phase to phase 0 and then to phase 4. Or you turn OFF - ON the drive
If everything is ok you get green light on the drive.

7. Motor commutation

Turn ON the power to the Power supply
Set TRUE on PowerSupplyCheckSet of the PSD. Set to 2 / test the MotorCommutationControl.

The motor should move and the commutation procedure start
At the end the MotorCommutationState gives to you the right feedback.

Turn to 0 the MotorCommutationControl and turn OFF the PowerSupplyCheckSet.