M262 / LXM32S (SercosIII) Programming Example

MC_MoveAbsolute

Table of Contents

Introduction

1

Overview

This chapter gives the introduction.

Contents of this chapter

This chapter contains the following topics:

Before You Begin

General

The products specified in this document have been tested under actual service conditions. Of course, your specific application requirements may be different from those assumed for this and any related examples described herein. In that case, you will have to adapt the information provided in this and other related documents to your particular needs. To do so, you will need to consult the specific product documentation of the hardware and/or software components that you may add or substitute for any examples specified in this documentation. Pay particular attention and conform to any safety information, different electrical requirements and normative standards that would apply to your adaptation.

⚠ WARNING

REGULATORY INCOMPATIBILITY

Be sure that all equipment applied and systems designed comply with all applicable local, regional and national regulations and standards

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material. A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved. Failure to observe this information can result in injury or equipment damage.

The use and application of the information contained herein require expertise in the design and programming of automated control systems. Only the user or integrator can be aware of all the conditions and factors present during installation and setup, operation, and maintenance of the machine or process, and can therefore determine the automation and associated equipment and the related safeties and interlocks which can be effectively and properly used. When selecting automation and control equipment, and any other related equipment or software, for a particular application, the user or integrator must also consider any applicable local, regional or national standards and/or regulations.

Some of the major software functions and/or hardware components used in the proposed architectures and examples described in this document cannot be substituted without significantly compromising the performance of your application. Further, any such substitutions or alterations may completely invalidate any proposed architectures, descriptions, examples, instructions, wiring diagrams and/or compatibilities between the various hardware components and software functions specified herein and in related documentation. You must be aware of the consequences of any modifications, additions or substitutions.

A residual risk, as defined by EN/ISO 12100-1, Article 5, will remain if
- it is necessary to modify the recommended logic and if the added or modified components are not properly integrated in the control circuit.
- you do not follow the required standards applicable to the operation of the machine, or if the adjustments to and the maintenance of the machine are not properly made (it is essential to strictly follow the prescribed machine maintenance schedule).
- the devices connected to any safety outputs do not have mechanically-linked contacts.

⚠ CAUTION
EQUIPMENT INCOMPATIBILITY Read and thoroughly understand all device and software documentation before attempting any component substitutions or other changes related to the application examples provided in the document **Failure to follow these instructions can result in injury, or equipment damage.**

Start-Up and Test

Before using electrical control and automation equipment after design and installation, the application and associated functional safety system must be subjected to a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such testing be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ CAUTION

EQUIPMENT OPERATION HAZARD

- Verify that all installation and setup procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in injury, or equipment damage.

Verify that the completed system, including the functional safety system, is free from all short circuits and grounds, except those grounds installed according to local regulations. If high-potential voltage testing is necessary, follow the recommendations in equipment documentation to help prevent injury or equipment damage.

Operations and Adjustments

General

Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly installed and operated.
In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the hands and other parts of the body are free to enter the pinch points or other hazardous areas where serious injury can occur. Software products alone cannot protect an operator from injury. For this reason, the software cannot be substituted for or take the place of point-of-operation protection.

⚠ WARNING
UNGUARDED MACHINERY CAN CAUSE SERIOUS INJURY •Do not use this software and related automation equipment on equipment which does not have point-of -operation protection. •Do not reach into machinery during operation. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

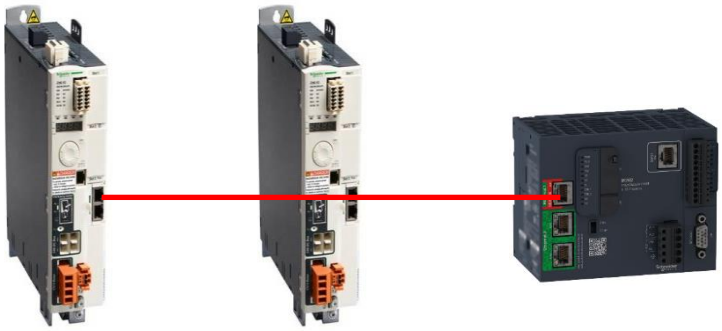
Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-ofoperation protection is outside the scope of the examples and implementations suggested herein. It is sometimes possible to adjust the equipment incorrectly and this produce unsatisfactory or unsafe operation. Always use the manufacturer instructions as a guide to functional adjustments. Personnel who have access to these adjustments must be familiar with the equipment manufacturer instructions and the machinery used with the electrical equipment. Only those operational adjustments actually required by the machine operator should be accessible to the operator. Access to other controls should be restricted to help prevent unauthorized changes in operating characteristics.

Functional overview

2

Components



The Motion Controller TM262Mxxxx controls 2 Lexium32S devices over SERCOS III.

Minimum required LXM32S firmware versions:
- LXM32S drive firmware PR912.20 Version >= 1.06.03
- LXM32S SERCOS module firmware PR940.00 Version >= 1.08.04

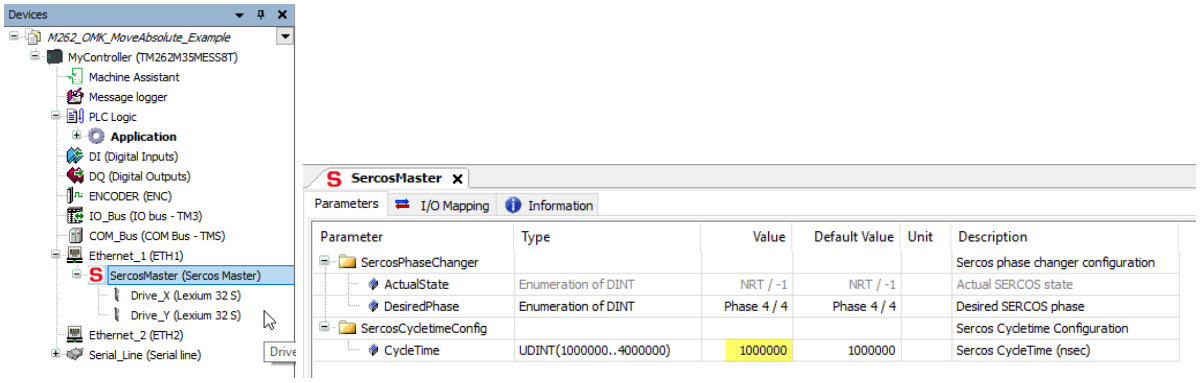
The example program contents a basic
- Axis configuration 2 Linear Axis without software limits
- Switching ON the power stage (MC_Power)
- Homing the axis (MC_Home)
- Move absolute 2 axis (MC_MoveAbsolute)

A detailed error handling is not part of this example.

SERCOS Configuration

3

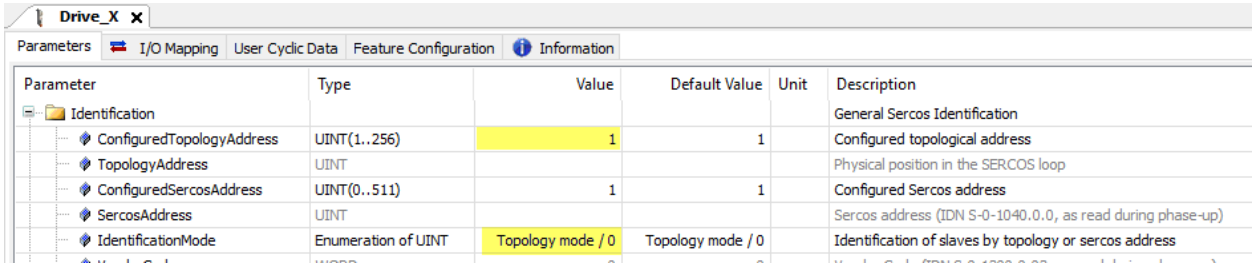
Sercos Master



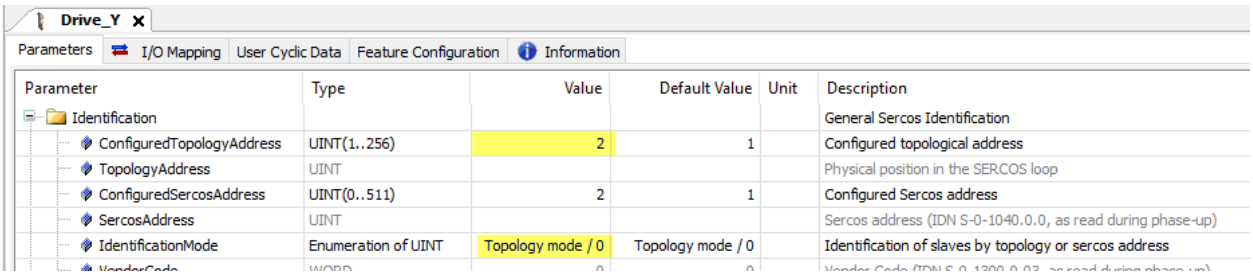
This example uses the DEFAULT value of 1 ms for the Sercos cycle time

Sercos Slaves

X-Axis – Topology address 1



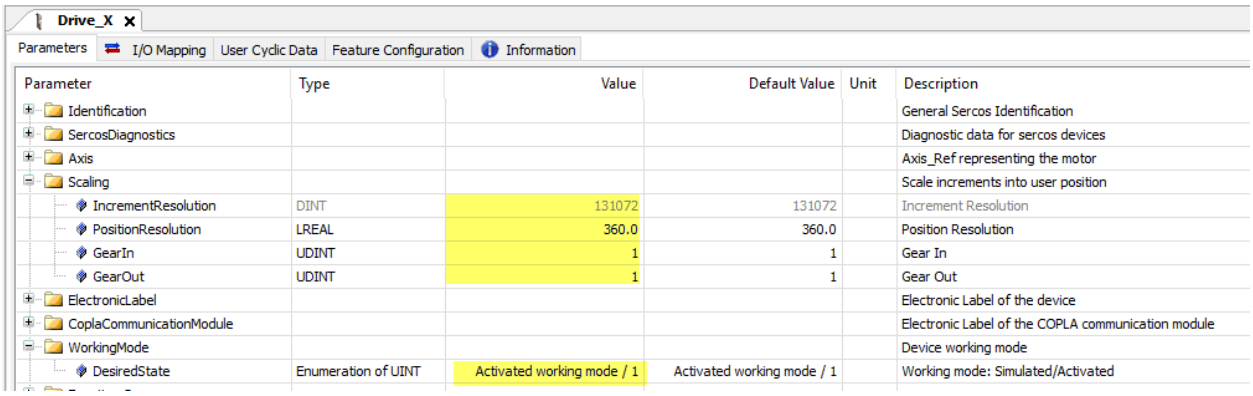
Y-Axis – Topology address 2



Axis configuration

Drive_X

Real axis with 360 user units / revolution
No mechanical gear (ratio 1:1)



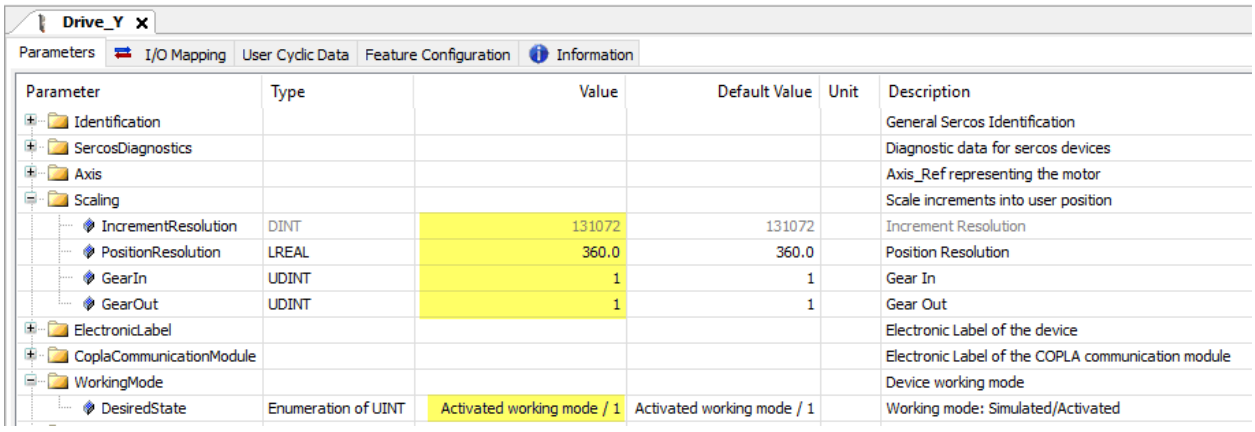
Drive_X

Parameters | I/O Mapping | User Cyclic Data | Feature Configuration | Information

Parameter	Type	Value	Default Value	Unit	Description
Identification					General Sercos Identification
SercosDiagnostics					Diagnostic data for sercos devices
Axis					Axis_Ref representing the motor
Scaling					Scale increments into user position
IncrementResolution	DINT	131072	131072		Increment Resolution
PositionResolution	LREAL	360.0	360.0		Position Resolution
GearIn	UDINT	1	1		Gear In
GearOut	UDINT	1	1		Gear Out
ElectronicLabel					Electronic Label of the device
CoplaCommunicationModule					Electronic Label of the COPLA communication module
WorkingMode					Device working mode
DesiredState	Enumeration of UINT	Activated working mode / 1	Activated working mode / 1		Working mode: Simulated/Activated

Drive_Y

Real axis with 360 user units / revolution
No mechanical gear (ratio 1:1)

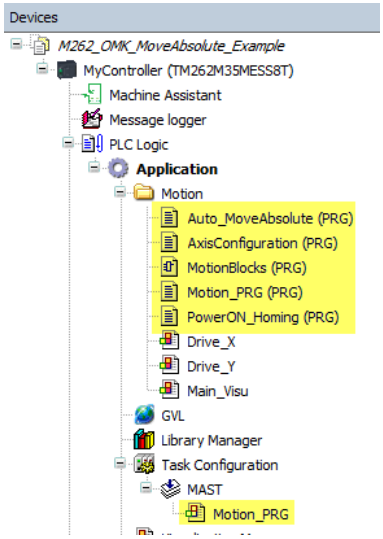


Drive_Y

Parameters | I/O Mapping | User Cyclic Data | Feature Configuration | Information

Parameter	Type	Value	Default Value	Unit	Description
Identification					General Sercos Identification
SercosDiagnostics					Diagnostic data for sercos devices
Axis					Axis_Ref representing the motor
Scaling					Scale increments into user position
IncrementResolution	DINT	131072	131072		Increment Resolution
PositionResolution	LREAL	360.0	360.0		Position Resolution
GearIn	UDINT	1	1		Gear In
GearOut	UDINT	1	1		Gear Out
ElectronicLabel					Electronic Label of the device
CoplaCommunicationModule					Electronic Label of the COPLA communication module
WorkingMode					Device working mode
DesiredState	Enumeration of UINT	Activated working mode / 1	Activated working mode / 1		Working mode: Simulated/Activated

POU's



Main program (MOTION_PRG)

Motion_PRG

```
PROGRAM Motion_PRG
VAR
END_VAR
```

```
// ********************************************************************************
// Initialization / Axis configuration
// ********************************************************************************
AxisConfiguration();
IF NOT GVL.gxInit_OK THEN // Configuration finished and SERCOS in PHASE 4 ?
    RETURN;
END_IF
// ********************************************************************************
// Cyclic call of Motion function block
MotionBlocks();
// ********************************************************************************
// Sequence to power ON and home the drives
PowerON_Homing();
// ********************************************************************************
// Call of automatic example sequence
Auto_MoveAbsolute();
// ********************************************************************************
```

AxisConfiguration (PRG)

- Setup axis type to Linear without SW limits
- Phase up SERCOS to CP4

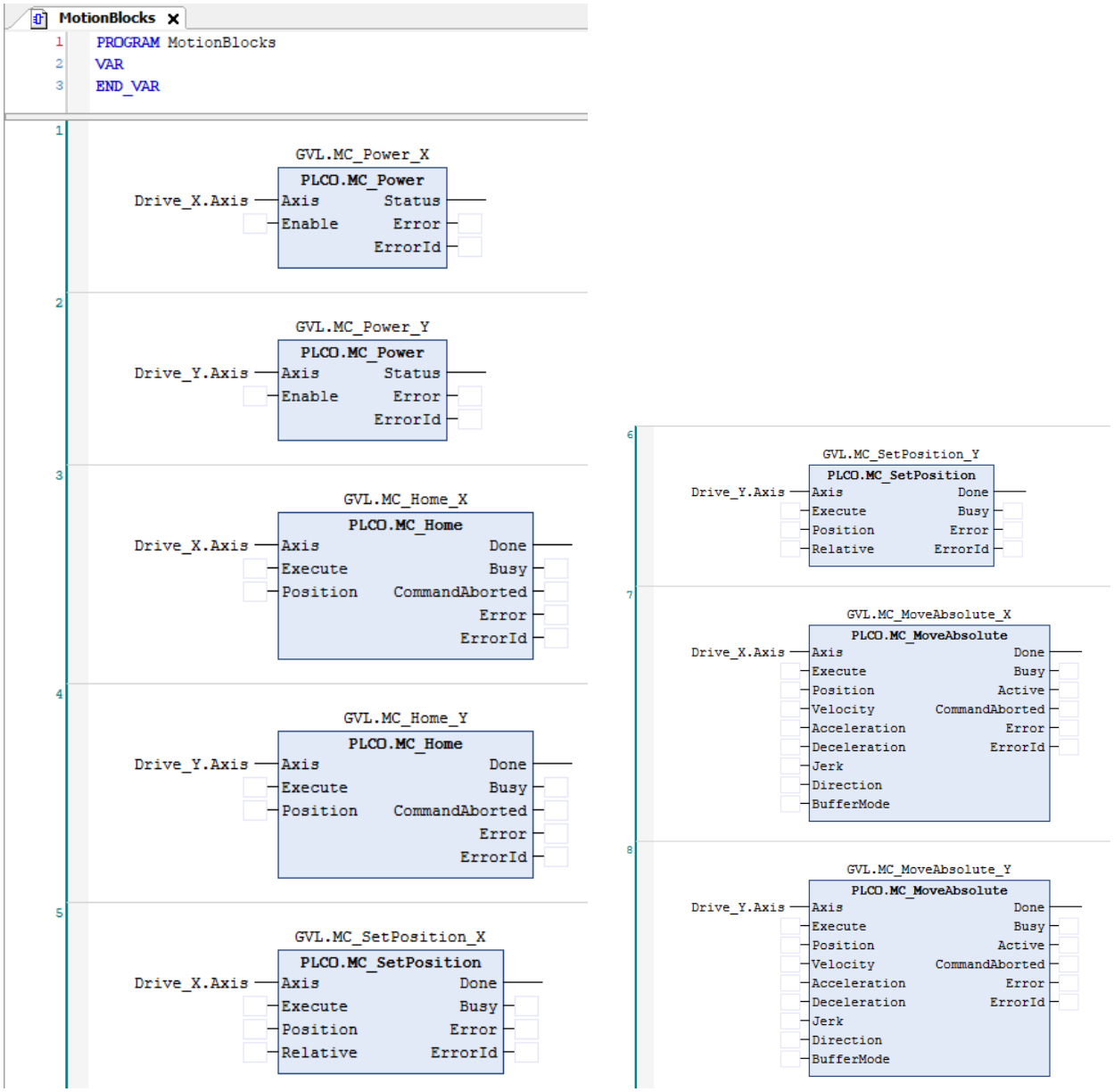
AxisConfiguration

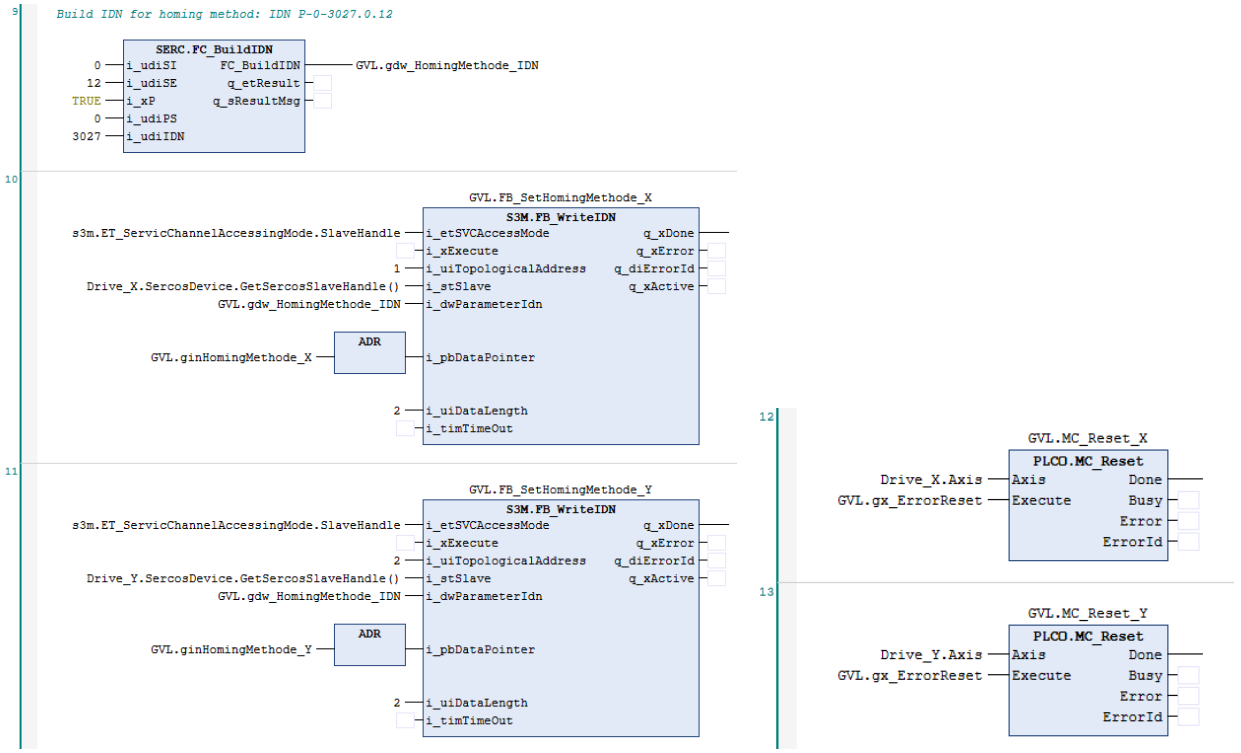
```
PROGRAM AxisConfiguration
VAR
    iState      :INT;
END_VAR
```

```
// ********************************************************************************************************************
// Axis configuration and SERCOS III phase up !!!
// ********************************************************************************************************************
CASE iState OF
    0:  // Set Sercos Phase to NRT (-1)
        GVL.gxInit_OK   :=FALSE;
        GVL.gSercosPhaseSet :=S3M.ET_SercosPhase.NRT;
        IF SercosMaster.SercosPhaseChanger.ActualState = S3M.ET_SercosState.NRT THEN
            iState  :=iState + 1;
        END_IF
    1:  // ************************************************************************
        // Axis Configuration
        // ************************************************************************
        // Axis type
        // Linear Axis without limits
        Drive_X.Axis.SetAxisTypeLinearWithoutLimits();
        Drive_Y.Axis.SetAxisTypeLinearWithoutLimits();
        iState  :=iState + 1;
    2:  // ************************************************************************
        // Set Sercos Phase 4
        // ************************************************************************
        GVL.gSercosPhaseSet := S3M.ET_SercosPhase.Phase4;
        IF SercosMaster.SercosPhaseChanger.ActualState = S3M.ET_SercosState.Phase4 THEN
            GVL.gxInit_OK   :=TRUE;
            iState  :=iState + 1;
        END_IF

    3:  // ************************************************************************
        // Check for New - Configuration Command
        // ************************************************************************
        IF NOT GVL.gxInit_OK THEN
            iState  :=0;    // New phase up
        END_IF
END_CASE
// ********************************************************************************************************************
// Check Sercos Phase 4
SercosMaster.SercosPhaseChanger.DesiredPhase    := GVL.gSercosPhaseSet;
IF SercosMaster.SercosPhaseChanger.ActualState = S3M.ET_SercosState.Phase4 THEN
    GVL.gx_SercosInPhase4   :=TRUE;
ELSE
    GVL.gx_SercosInPhase4   :=FALSE;
END_IF
// ********************************************************************************************************************
```

MotionBlocks (PRG)

- Cyclic call of PLCopen function blocks and other motion function blocks




9
Build IDN for homing method: IDN P-0-3027.0.12
SERC.FC_BuildIDN
0 i_udiSI
12 i_udiSE
TRUE i_xP
0 i_udiPS
3027 i_udiIDN
FC_BuildIDN GVL.gdw_HomingMethode_IDN
q_etResult
q_sResultMsg
10
GVL.FB_SetHomingMethode_X
S3M.FB_WriteIDN
s3m.ET_ServicChannelAccessingMode.SlaveHandle i_etSVCAccessMode
i_xExecute
1 i_uiTopologicalAddress
Drive_X.SercosDevice.GetSercosSlaveHandle() i_stSlave
GVL.gdw_HomingMethode_IDN i_dwParameterIdn
GVL.ginHomingMethode_X ADR i_pbDataPointer
2 i_uiDataLength
i_timTimeOut
q_xDone
q_xError
q_diErrorId
q_xActive
11
GVL.FB_SetHomingMethode_Y
S3M.FB_WriteIDN
s3m.ET_ServicChannelAccessingMode.SlaveHandle i_etSVCAccessMode
i_xExecute
2 i_uiTopologicalAddress
Drive_Y.SercosDevice.GetSercosSlaveHandle() i_stSlave
GVL.gdw_HomingMethode_IDN i_dwParameterIdn
GVL.ginHomingMethode_Y ADR i_pbDataPointer
2 i_uiDataLength
i_timTimeOut
q_xDone
q_xError
q_diErrorId
q_xActive
12
GVL.MC_Reset_X
PLCO.MC_Reset
Drive_X.Axis Axis Done
GVL.gx_ErrorReset Execute Busy
Error
ErrorId
13
GVL.MC_Reset_Y
PLCO.MC_Reset
Drive_Y.Axis Axis Done
GVL.gx_ErrorReset Execute Busy
Error
ErrorId

PowerON_Homing (PRG)

PowerON_Homing

```
PROGRAM PowerON_Homing
VAR
    iState      :INT;
    iErrorState :INT;
END_VAR
```

```
// ************************************************************
// Reset state machine if "Homing command = FALSE"
// ************************************************************
IF NOT GVL.gx_PowerON_Homing AND iState >0 THEN
    GVL.MC_Home_X.Execute   :=FALSE;
    GVL.MC_Home_Y.Execute   :=FALSE;
    GVL.MC_Power_X.Enable   :=FALSE;
    GVL.MC_Power_Y.Enable   :=FALSE;
    GVL.FB_SetHomingMethode_X.i_xExecute    :=FALSE;
    GVL.FB_SetHomingMethode_Y.i_xExecute    :=FALSE;
    GVL.gx_Homing_OK        :=FALSE;
    iState  :=0;
END_IF
CASE iState OF
    0:  // ********************************************************************************************
        // Wait for start command -> Power ON
        // ********************************************************************************************
        IF GVL.gx_PowerON_Homing THEN       // Start Command from VISU Button
            GVL.gx_PowerON_HomingBusy   :=TRUE;     // Set Sequence is BUSY (active)
            GVL.MC_Power_X.Enable       :=TRUE;     // Enable power stage Drive X
            GVL.MC_Power_Y.Enable       :=TRUE;     // Enable power stage Drive Y
            IF GVL.MC_Power_X.Status AND GVL.MC_Power_Y.Status THEN // power stage enabled
                iState  :=iState + 1;
            ELSIF GVL.MC_Power_X.Error OR GVL.MC_Power_Y.Error THEN // Error
                iErrorState :=iState;
                iState      :=100;
            END_IF
        END_IF
    1:  // ********************************************************************************************
        // Set Homing Type P-0-3027.0.12 - Drive X
        // 17 = Limit Switch negative
        // 33 = Index pulse
        // ...
        // ********************************************************************************************
        GVL.ginHomingMethode_X  :=33;//17;
        GVL.FB_SetHomingMethode_X.i_xExecute    :=TRUE;
        IF GVL.FB_SetHomingMethode_X.q_xDone THEN
            GVL.FB_SetHomingMethode_X.i_xExecute    :=FALSE;
            iState  :=iState + 1;
        ELSIF GVL.FB_SetHomingMethode_X.q_xError THEN
            iState      :=100;
        END_IF
    2:  // ********************************************************************************************
        // Set Homing Type P-0-3027.0.12 - Drive Y
        // 17 = Limit Switch negative
        // 33 = Index pulse
        // ...
        // ********************************************************************************************
        GVL.ginHomingMethode_Y  :=33;//17;
        GVL.FB_SetHomingMethode_Y.i_xExecute    :=TRUE;
        IF GVL.FB_SetHomingMethode_Y.q_xDone THEN
            GVL.FB_SetHomingMethode_Y.i_xExecute    :=FALSE;
            iState  :=iState + 1;
        ELSIF GVL.FB_SetHomingMethode_Y.q_xError THEN
            iState      :=100;
        END_IF
```

```
    3:  // ********************************************************************************
        // Homing Drive X and Y
        // ********************************************************************************
        GVL.MC_Home_X.Execute   :=TRUE;
        GVL.MC_Home_X.Position  := 0;
        GVL.MC_Home_Y.Execute   :=TRUE;
        GVL.MC_Home_Y.Position  := 0;
        IF GVL.MC_Home_X.Done AND GVL.MC_Home_Y.Done THEN   // Homing completed X + Y
            GVL.MC_Home_X.Execute   :=FALSE;
            GVL.MC_Home_Y.Execute   :=FALSE;
            iState  :=iState + 1;
        ELSIF GVL.MC_Home_X.Error OR GVL.MC_Home_X.CommandAborted OR GVL.MC_Home_Y.Error OR GVL.MC_Home_Y.CommandAborted THEN
            iState  :=100;
        END_IF
    4:  // ********************************************************************************
        // Homing completed
        // ********************************************************************************
        GVL.gx_PowerON_HomingBusy   :=FALSE;
        GVL.gx_Homing_OK            :=TRUE;         // Homing OK -> ready for automatic sequence
        IF NOT GVL.gx_PowerON_Homing THEN   // New command to start this sequence
            GVL.MC_Power_X.Enable       :=FALSE;
            GVL.MC_Power_Y.Enable       :=FALSE;
            GVL.gx_Homing_OK            :=FALSE;
            iState  :=0;
        END_IF
    100:// ********************************************************************************
        // Abort/Error State
        // ********************************************************************************
        ;
END_CASE
```

Auto_MoveAbsolute (PRG)

Automatic example sequence where the axis moved by the FB MC_MoveAbsolute

Auto_MoveAbsolute

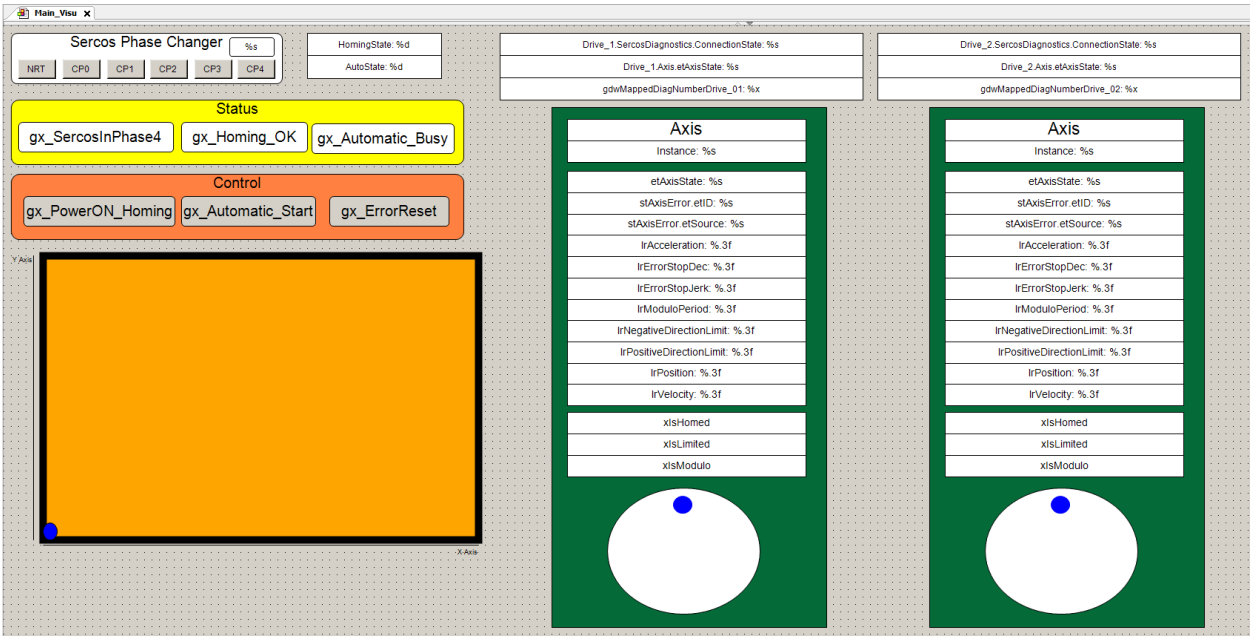
```
PROGRAM Auto_MoveAbsolute
VAR
    iState          :INT;
END_VAR
```

```
// ********************************************************************************
// Reset state machine if "AutomaticStart = FALSE"
// ********************************************************************************
IF NOT GVL.gx_Automatic_Start AND iState >0 THEN
    GVL.MC_MoveAbsolute_X.Execute   :=FALSE;
    GVL.MC_MoveAbsolute_Y.Execute   :=FALSE;
    iState  :=0;
END_IF
```

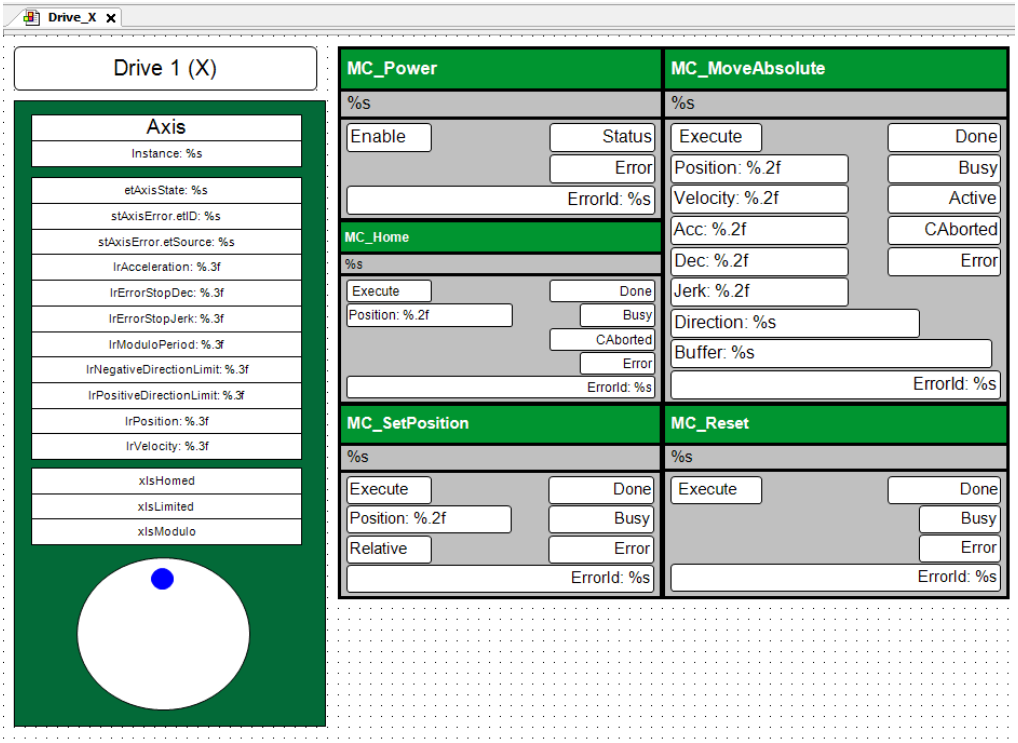
```
CASE iState OF
    0:  // ***************************************************************
        // Wait for START command
        // ***************************************************************
        IF GVL.gx_Automatic_Start AND GVL.gx_Homing_OK THEN
            GVL.gx_Automatic_Busy               :=TRUE;
            // Drive X
            GVL.MC_MoveAbsolute_X.Position      :=800;
            GVL.MC_MoveAbsolute_X.Velocity      :=1500;
            GVL.MC_MoveAbsolute_X.Acceleration  :=1500;
            GVL.MC_MoveAbsolute_X.Deceleration  :=1500;
            GVL.MC_MoveAbsolute_X.Execute       :=TRUE;
            // Drive Y
            GVL.MC_MoveAbsolute_Y.Position      :=350;
            GVL.MC_MoveAbsolute_Y.Velocity      :=1500;
            GVL.MC_MoveAbsolute_Y.Acceleration  :=1500;
            GVL.MC_MoveAbsolute_Y.Deceleration  :=1500;
            GVL.MC_MoveAbsolute_Y.Execute       :=TRUE;
            iState  :=iState + 1;
        ELSE
            GVL.gx_Automatic_Busy               :=FALSE;
        END_IF
    1:  // ***************************************************************************************
        // Wait for target position reached
        // ***************************************************************************************
        IF GVL.MC_MoveAbsolute_X.Done AND GVL.MC_MoveAbsolute_Y.Done THEN
            GVL.MC_MoveAbsolute_X.Execute   :=FALSE;
            GVL.MC_MoveAbsolute_Y.Execute   :=FALSE;
            iState  :=iState + 1;
        ELSIF GVL.MC_MoveAbsolute_X.Error OR GVL.MC_MoveAbsolute_X.CommandAborted
            OR GVL.MC_MoveAbsolute_Y.Error OR GVL.MC_MoveAbsolute_Y.CommandAborted THEN
            iState      :=100;
        END_IF
    2:  // ***************************************************************************************
        // Move Drive Y to position 0
        // ***************************************************************************************
        GVL.MC_MoveAbsolute_Y.Position  :=0;
        GVL.MC_MoveAbsolute_Y.Execute   :=TRUE;
        IF GVL.MC_MoveAbsolute_Y.Done THEN  // target pos reached
            GVL.MC_MoveAbsolute_Y.Execute   :=FALSE;
            iState  :=iState + 1;
        ELSIF GVL.MC_MoveAbsolute_Y.Error OR GVL.MC_MoveAbsolute_Y.CommandAborted THEN
            iState      :=100;
        END_IF
    3:  // ***************************************************************************************
        // Move Drive X to position 0
        // ***************************************************************************************
        GVL.MC_MoveAbsolute_X.Position  :=0;
        GVL.MC_MoveAbsolute_X.Execute   :=TRUE;
        IF GVL.MC_MoveAbsolute_X.Done THEN  // target pos reached
            GVL.MC_MoveAbsolute_X.Execute   :=FALSE;
            iState  :=0;                    // state 0 -> repeat this sequence
        ELSIF GVL.MC_MoveAbsolute_X.Error OR GVL.MC_MoveAbsolute_X.CommandAborted THEN
            iState      :=100;
        END_IF

    100:// ***************************************************************************************
        // Abort / Error State
        // ***************************************************************************************
        GVL.gx_Automatic_Busy   :=FALSE;
END_CASE
```

Visualizations

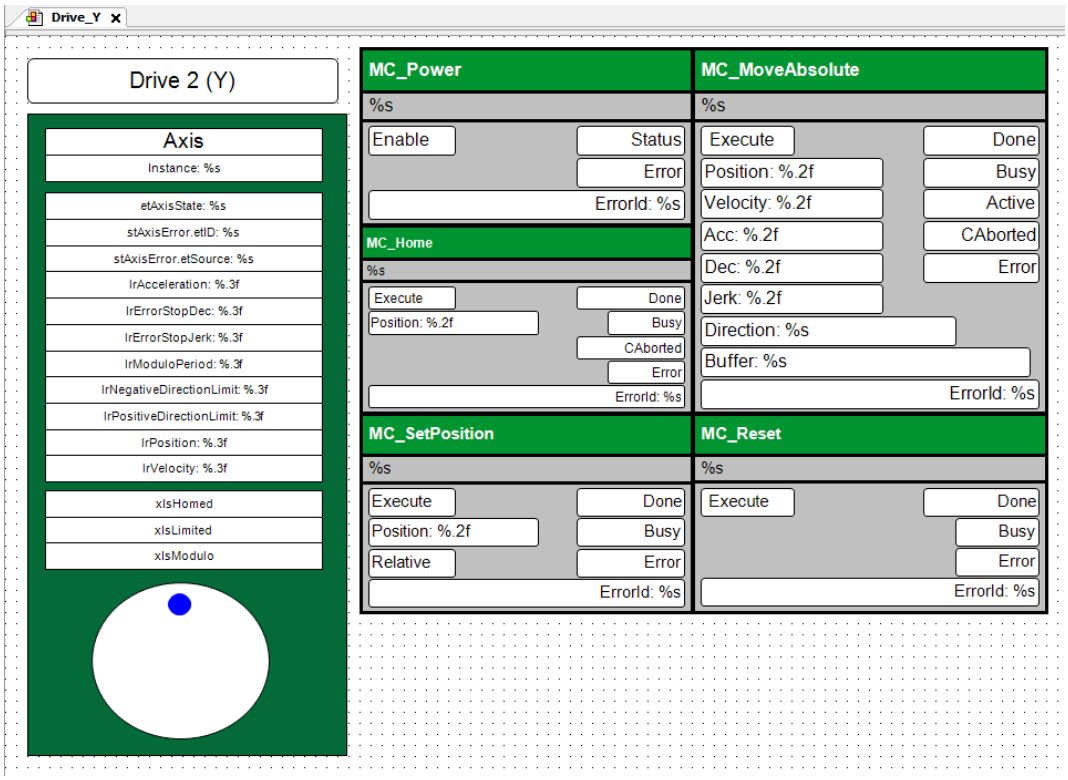
Main_VISU



DRIVE_X



DRIVE_Y


Drive_Y
Drive 2 (Y)
Axis
Instance: %s
etAxisState: %s
stAxisError.etID: %s
stAxisError.etSource: %s
lrAcceleration: %.3f
lrErrorStopDec: %.3f
lrErrorStopJerk: %.3f
lrModuloPeriod: %.3f
lrNegativeDirectionLimit: %.3f
lrPositiveDirectionLimit: %.3f
lrPosition: %.3f
lrVelocity: %.3f
xIsHomed
xIsLimited
xIsModulo
MC_Power
%s
Enable
Status
Error
ErrorId: %s
MC_MoveAbsolute
%s
Execute
Done
Position: %.2f
Busy
Velocity: %.2f
Active
Acc: %.2f
CAborted
Dec: %.2f
Error
Jerk: %.2f
Direction: %s
Buffer: %s
ErrorId: %s
MC_Home
%s
Execute
Done
Position: %.2f
Busy
CAborted
Error
ErrorId: %s
MC_SetPosition
%s
Execute
Done
Position: %.2f
Busy
Relative
Error
ErrorId: %s
MC_Reset
%s
Execute
Done
Busy
Error
ErrorId: %s