DOCUMENTATION ISG-kernel

PLC library
ISG Motion Control Platform for PLCopen

Short Description:
MCP-INTRO

Documentation version: 1.03
13/12/2023

Table of contents

List of figures

1 Definitions

1.1 Abbreviations

AXHLI	Axis-specific High-Level Interface
CM	Continuous Motion (endless rotation)
DM	Discrete Motion (positioning)
FB	Function Block
FBSD	FB State Diagram
HLI	High-Level Interface between MC and PLC
MC	Motion Controller
MCP	Motion Control Platform
MCE	Motion Control Engine
MC-FB	Motion Controller Function Block
NL Slope	Non-linear slope
PCS	Part program coordinate system
PLC	Programmable Logic Control
POE	Program Organisation Unit
SAI	Single Axis Interpolator

1.2 Explanations of terms

Axis group	A combination of axes which can execute a motion on a spatial curve coordinated by a channel while maintaining the specified values for velocity, acceleration and jerk on this spatial curve.
CoDeSys	PLC programming system from 3S Smart Software Solutions
Function block:	Internal order format of the ISG Motion Controller.
HLI library	Access to the memory interface to the ISG-MCE.
ISG-MCE	This stands for the ISG NC Kernel which, in connection with this documentation, is also referred to as the "Motion Control Engine"
Channel	Unit which coordinates the axis motions of an axis group.
MC-FB	Designates the PLC function blocks that are used to issue commands to the ISG-MC.
Multiprog	PLC programming system from KW-Software
Motion library	PLC software application that contains function blocks to move axes in conformity with the PLCopen specification as well as further FBs to assume motion generation tasks

Axis group	A combination of axes which can execute a motion on a spatial curve coordinated by a channel while maintaining the specified values for velocity, acceleration and jerk on this spatial curve.
CoDeSys	PLC programming system from 3S Smart Software Solutions
Function block:	Internal order format of the ISG Motion Controller.
HLI library	Access to the memory interface to the ISG-MCE.
ISG-MCE	This stands for the ISG NC Kernel which, in connection with this documentation, is also referred to as the "Motion Control Engine"
Channel	Unit which coordinates the axis motions of an axis group.
MC-FB	Designates the PLC function blocks that are used to issue commands to the ISG-MC.
Motion library	PLC software application that contains function blocks to move axes in conformity with the PLCopen specification as well as further FBs to assume motion generation tasks

Axis group	A combination of axes which can execute a motion on a spatial curve coordinated by a channel while maintaining the specified values for velocity, acceleration and jerk on this spatial curve.
Function block:	Internal order format of the ISG Motion Controller.
HLI library	Access to the memory interface to the ISG-MCE.
ISG-MCE	This stands for the ISG NC Kernel which, in connection with this documentation, is also referred to as the "Motion Control Engine"
Channel	Unit which coordinates the axis motions of an axis group.
MC-FB	Designates the PLC function blocks that are used to issue commands to the ISG-MC.
Multiprog	PLC programming system from KW-Software
Motion library	PLC software application that contains function blocks to move axes in conformity with the PLCopen specification as well as further FBs to assume motion generation tasks

Mandatory note on references to other documents

For the sake of clarity, links to other documents and parameters are abbreviated, e.g. [PROG] for the Programming Manual or P-AXIS-00001 for an axis parameter.

For technical reasons, these links only function in the Online Help (HTML5, CHM) but not in pdf files since pdfs do not support cross-linking.

2 The ISG Motion Control Platform

2.1 What is the ISG Motion Control Platform?

The ISG-MCP is a PLC library that can also be delivered in IEC 61131 source if requested by customers. It enables PLC application programmers to program motion tasks in accordance to the PLCopen specification within an IEC 61131 PLC. All the functions required internally to generate motion are hidden from the PLC application programmer, e.g.:

- Interpolation
- Position controller
- operating drive interfaces etc.

The ISG-MCP provides the function blocks, data structures and state models defined in the PLCopen specification [1].

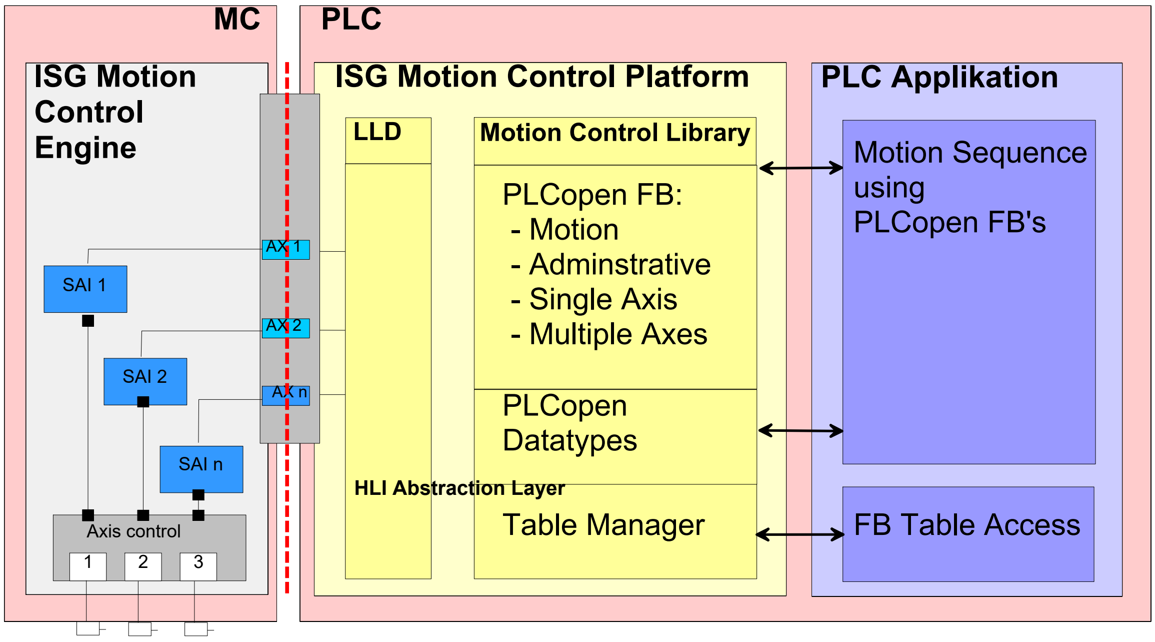


Fig. 1: The PLC application programmer sees the ISG-MCP as one single programming interface.

2.2 Elements of the ISG Motion Control Platform

The ISG Motion Control Platform consists of various PLC user libraries. They contain: FBs and data types according to PLCopen specifications as well as those specified by controller manufacturers. These elements are easily distinguished by their prefix:

- All elements identified by the prefix **MC_** are listed in the various parts of the PLCopen specifications.
- The elements with the prefix **MCV_**are specified by the controller manufacturer.

2.2.1 HLI library – memory interface to the ISG-MCE

A component of the ISG-MCP is the hli.lib user library.

It contains the definition of the memory interface HLI to the ISG-MCE. The PLCopen FB sends commands over this interface to move its assigned axes and receive ISG-MCE axis-related messages.

For access to the HLI, the **hli** global variable is created in the ISG-MCP as the %M3.xxx variable in the Multiprog PLC environment.

A PLC application in the CoDeSys environment must invoke an instance of the FB MCV_HliInterface as the very first block to initialise the global pointer to access HLI areas (see Frame_PLCopenP1).

Notice

After successful initialisation, programs and function blocks of the user libraries described below may be used.

2.2.2 Platform library

The McpBase.lib user library contains definitions of data structures that represent objects as references in conformity with the PLCopen specification. Their use is intended to release motion tasks.

The references are already included in the library as global variables.

The variables must be created as global variables in the PLC application.

Another key component of the library is the FB **MCV_PlatformBase,** which must be used in every PLC application that releases motion tasks based on PLCopen specifications.

This FB assumes the task to initialise the reference structures and checks the consistency of the HLI interface on the PLC and MCE sides. Only when this FB sets the “Done” output to TRUE can motion commands be sent to the MC successfully using the FB of the next motion library listed (see Frame_PLCopenP1).

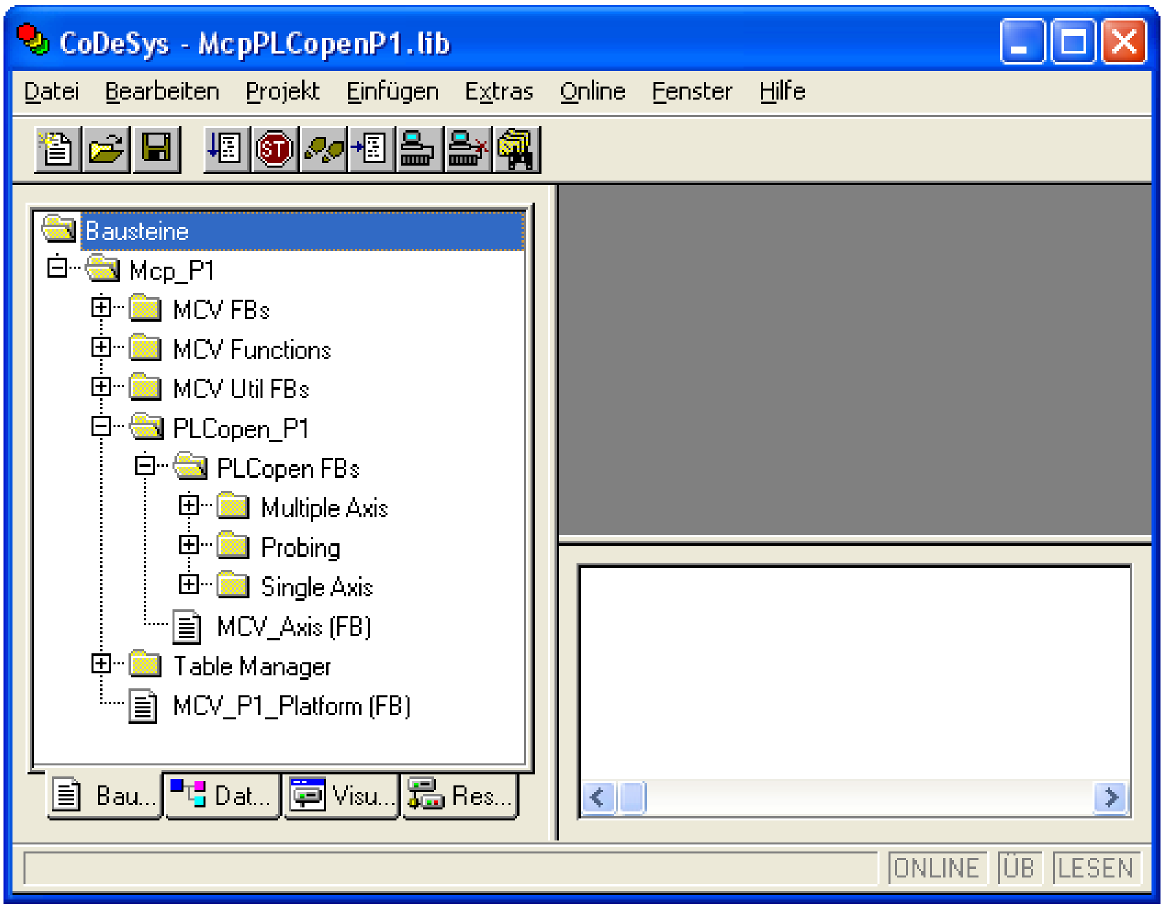
2.2.3 Motion library – PLCopen Part1

Besides FBs that comply with PLCopen specifications, the McpPLCopenP1.lib user library also defines FBs which cover additional functions and which must be used to implement an application. This library is referred to below as the motion library.

Release Note

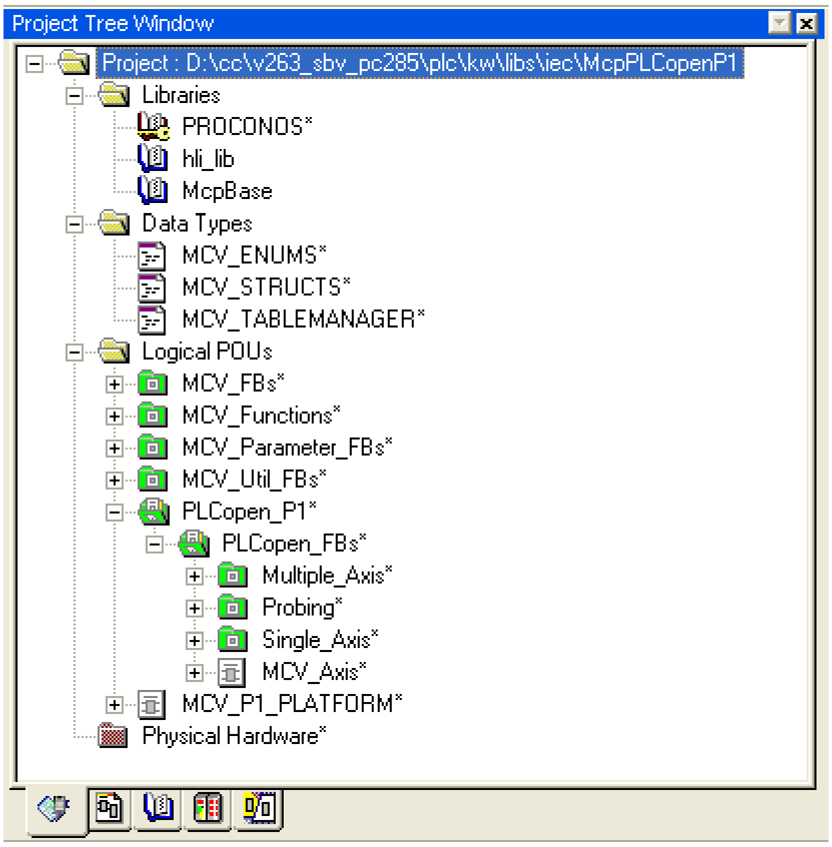
The scope of functions in the version varies depending on the PLC platform used.

The figure below shows the structure of the motion library. The main elements in the library are then explained in greater detail:


CoDeSys - McpPLCopenP1.lib
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Bau... Dat... Visu... Res...
ONLINE ÜB LESEN

```
Bausteine
  Mcp_P1
    MCV FBs
    MCV Functions
    MCV Util FBs
    PLCopen_P1
      PLCopen FBs
        Multiple Axis
        Probing
        Single Axis
      MCV_Axis (FB)
    Table Manager
    MCV_P1_Platform (FB)
```

Fig. 2: Overview of the McpPLCopenP1.lib motion library in CoDeSys


Project Tree Window

Project : D:\cc\v263_sbv_pc285\plc\kw\libs\iec\McpPLCopenP1
- Libraries
  - PROCONOS*
  - hli_lib
  - McpBase
- Data Types
  - MCV_ENUMS*
  - MCV_STRUCTS*
  - MCV_TABLEMANAGER*
- Logical POUs
  - MCV_FBs*
  - MCV_Functions*
  - MCV_Parameter_FBs*
  - MCV_Util_FBs*
  - PLCopen_P1*
    - PLCopen_FBs*
      - Multiple_Axis*
      - Probing*
      - Single_Axis*
      - MCV_Axis*
  - MCV_P1_PLATFORM*
- Physical Hardware*

Fig. 3: Structure of the McpPLCopenP1.zwt motion library in Multiprog

2.2.3.1 PLCopen function blocks

PLCopen specification Part1 subdivides the FBs defined there according to their usage into:

- administrative and
- motion-related FBs.

Within these two areas a further distinction is made depending on the application, i.e.:

- a single axis or
- multiple axes

The table below is organised accordingly and lists the function blocks according to the PLCopen specification Part1.

Notice

The FBs in italics and marked by an asterisk * are not implemented in the motion library Part1. However, the libraries may contain FBs which have a similar functionality but are specified by the controller manufacturer.

Subdivision of PLCopen FB Part1 into administrative and motion-related FBs

Administrative		**Motion**	
Single Axis	**Multiple Axis**	**Single Axis**	**Multiple Axis**
MC_Power	MC_CamTableSelect	MC_MoveAbsolute	MC_CamIn
MC_ReadStatus		MC_MoveRelative	MC_CamOut
MC_ReadAxisError		MC_MoveAdditive	MC_GearIn
MC_ReadParameter		MC_MoveSuperimposed	MC_GearOut
*MC_ReadBoolParameter**		MC_MoveVelocity	MC_Phasing
MC_WriteParameter		MC_Home	*MC_GearInPos**
*MC_WriteBoolParameter**		MC_Stop	
MC_ReadActualPosition		*MC_PositionProfile**	
MC_Reset		*MC_VelocityProfile**	
MC_TouchProbe		*MC_AccelerationProfile**	
MC_AbortTrigger		*MC_TorqueControl**	
*MC_ReadDigitalInput**		*MC_MoveContinuous**	
*MC_ReadDigitalOutput**		MC_Halt	
*MC_WriteDigitalOutput**			
MC_SetPosition			
MC_SetOverride			
*MC_ReadActualVelocity**			
*MC_ReadActualTorque**			
*MC_DigitalCamSwitch**			

2.2.3.2 Function block MCV_Axis

The FB MCV_Axis, which has an AXIS_REF structure as input/output variable, updates the data of an AXIS_REF structure. This FB also performs the following tasks:

- Logging an axis to the MCE via the HLI. This is done by setting the "plc_present_w" flag in the axis -specific HLI area
- Register the PLC via the HLI so that the PLC can command the MCE with spindle reset, controller enable, feed enable and drive ON for a specific axis.
- During initialisation, the consistency of the HLI is verified by checking the version identifier and the size of the HLI.
- Adoption of error messages reported by the MCE for each axis

In each PLC application which uses the PLCopen Part1 FB of the ISG-MCP, an instance of this FB must be created for each axis used and an AXIS_REF structure in the form of **g_array_axis_ref[i]** must be assigned to it as a VAR_IN_OUT parameter.

To ensure this, ISG-MCP contains the FB MCV_P1_PLATFORM (see Sec. MCV_P1_PLATFORM function block [▸ 13]) which must be called in a program of a PLC application. This ensures that the working data of one axis is updated in every PLC cycle.

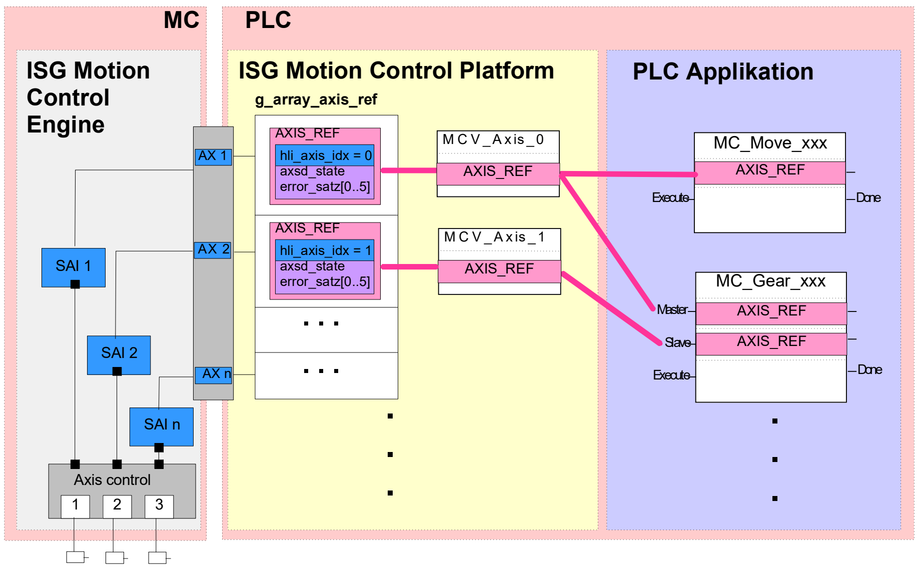


Fig. 4: Providing AXIS_REF by the FB “MCV_Axis”

Programing Example

Declaration and call in ST:

Declaration in ST:

```
cam_in_1   : MC_CamIn;
```

Call in ST:

```
cam_in_1   (Master:= g_array_axis_ref[0], Slave := g_array_axis_ref [1]);
```

2.2.3.3 Function block MCV_P1_PLATFORM

The following definition was made for the MCP:

Notice

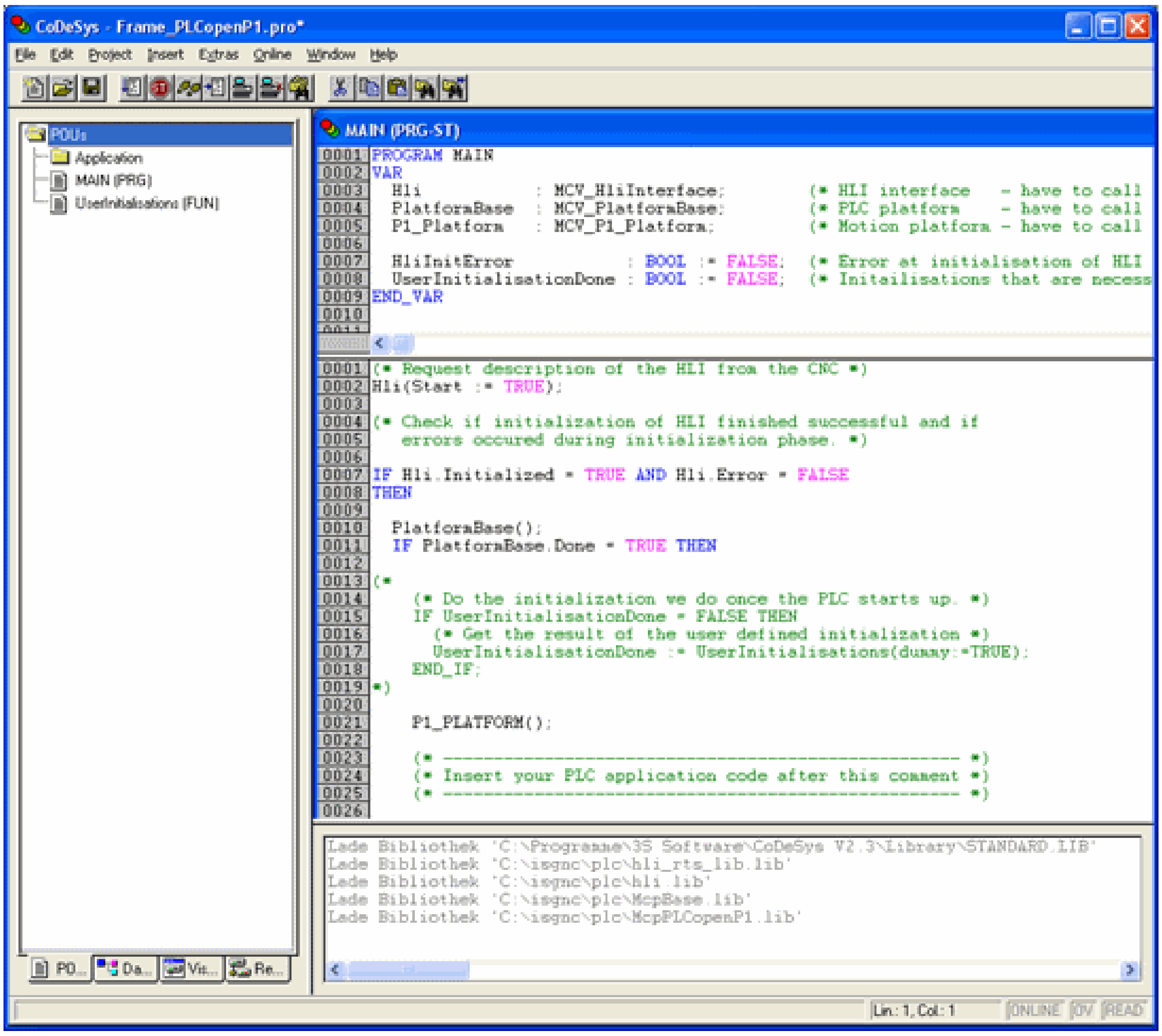
Definition for the MCP:

a) The generic FB "MCV_Axis" of PLCopen axes are instanced in the ISG-MCP and implemented in the FB MCV_P1_PLATFORM.

b) Every PLC application which executes motion tasks using FBs according to PLCopen specifications Parts 1 and 2 must contain exactly one instance of the FB MCV_P1_PLATFORM before the FBs to execute the motion task can be calculated.

c) Application programmers must ensure that all PLCopen FBs used to program the application (e.g. motion sequence) are instanced and invoked in a single application program.

d) Before the first invocation of the MCV_P1_PLATFORM instance, the HLI (interface to the MC) must be initialised and the instance of the FB MCV_PlatformBase must notify the MCP of successful initialisation.

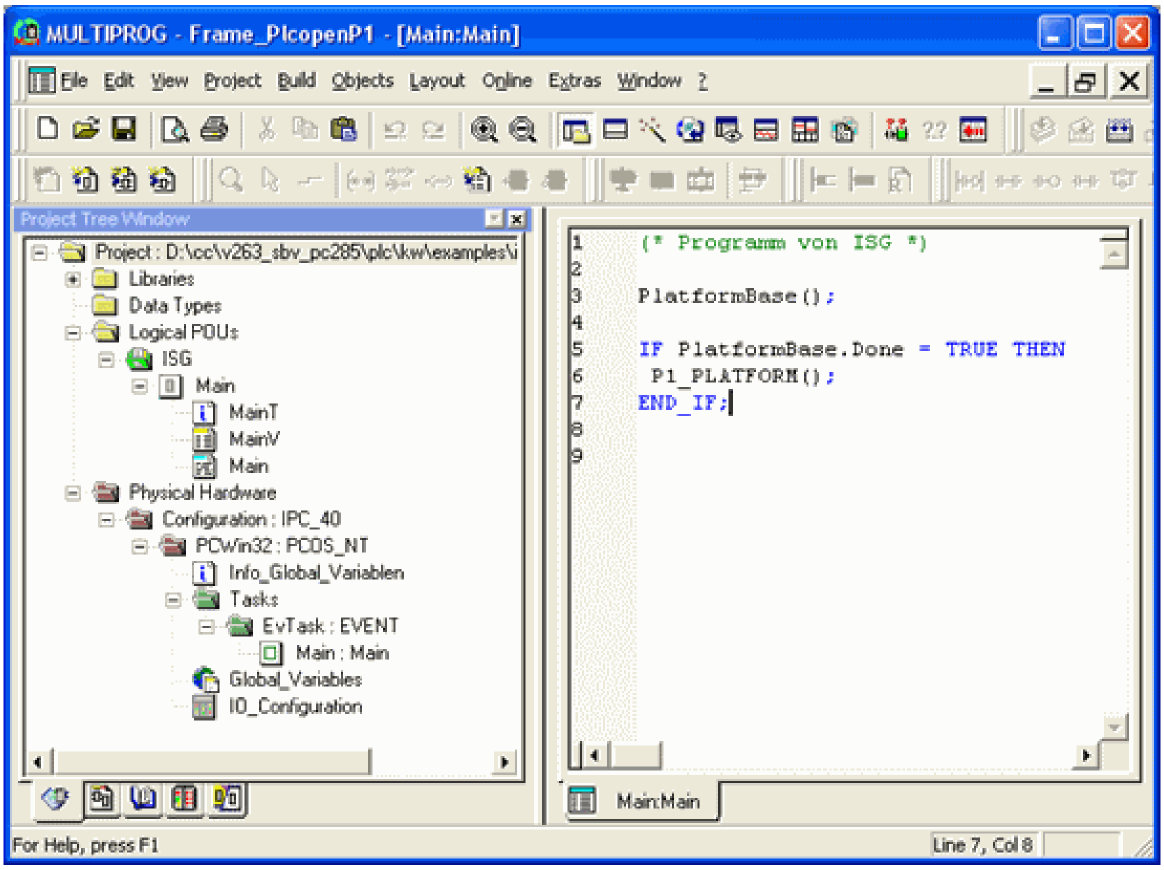

CoDeSys - Frame_PLCopenP1.pro*
File Edit Project Insert Extras Online Window Help
POUs
Application
MAIN (PRG)
UserInitialisations (FUN)
PO... Da... Vi... Re...
MAIN (PRG-ST)
Lin.: 1, Col.: 1
ONLINE OV READ

```
PROGRAM MAIN
VAR
  Hli                : MCV_HliInterface;          (* HLI interface    - have to call
  PlatformBase       : MCV_PlatformBase;          (* PLC platform     - have to call
  P1_Platform        : MCV_P1_Platform;           (* Motion platform - have to call

  HliInitError                 : BOOL := FALSE;   (* Error at initialisation of HLI
  UserInitialisationDone : BOOL := FALSE;   (* Initailisations that are necess
END_VAR
```

```
(* Request description of the HLI from the CNC *)
Hli(Start := TRUE);

(* Check if initialization of HLI finished successful and if
   errors occured during initialization phase. *)

IF Hli.Initialized = TRUE AND Hli.Error = FALSE
THEN

  PlatformBase();
  IF PlatformBase.Done = TRUE THEN

(*
    (* Do the initialization we do once the PLC starts up. *)
    IF UserInitialisationDone = FALSE THEN
      (* Get the result of the user defined initialization *)
      UserInitialisationDone := UserInitialisations(dummy:=TRUE);
    END_IF;
*)

    P1_PLATFORM();

    (* ---------------------------------------------------- *)
    (* Insert your PLC application code after this comment *)
    (* ---------------------------------------------------- *)
```

```
Lade Bibliothek 'C:\Programme\3S Software\CoDeSys V2.3\Library\STANDARD.LIB'
Lade Bibliothek 'C:\isgnc\plc\hli_rts_lib.lib'
Lade Bibliothek 'C:\isgnc\plc\hli.lib'
Lade Bibliothek 'C:\isgnc\plc\McpBase.lib'
Lade Bibliothek 'C:\isgnc\plc\McpPLCopenP1.lib'
```

Fig. 5: PLC application frame for motion applications in CoDeSys



```
(* Programm von ISG *)

PlatformBase();

IF PlatformBase.Done = TRUE THEN
 P1_PLATFORM();
END_IF;
```

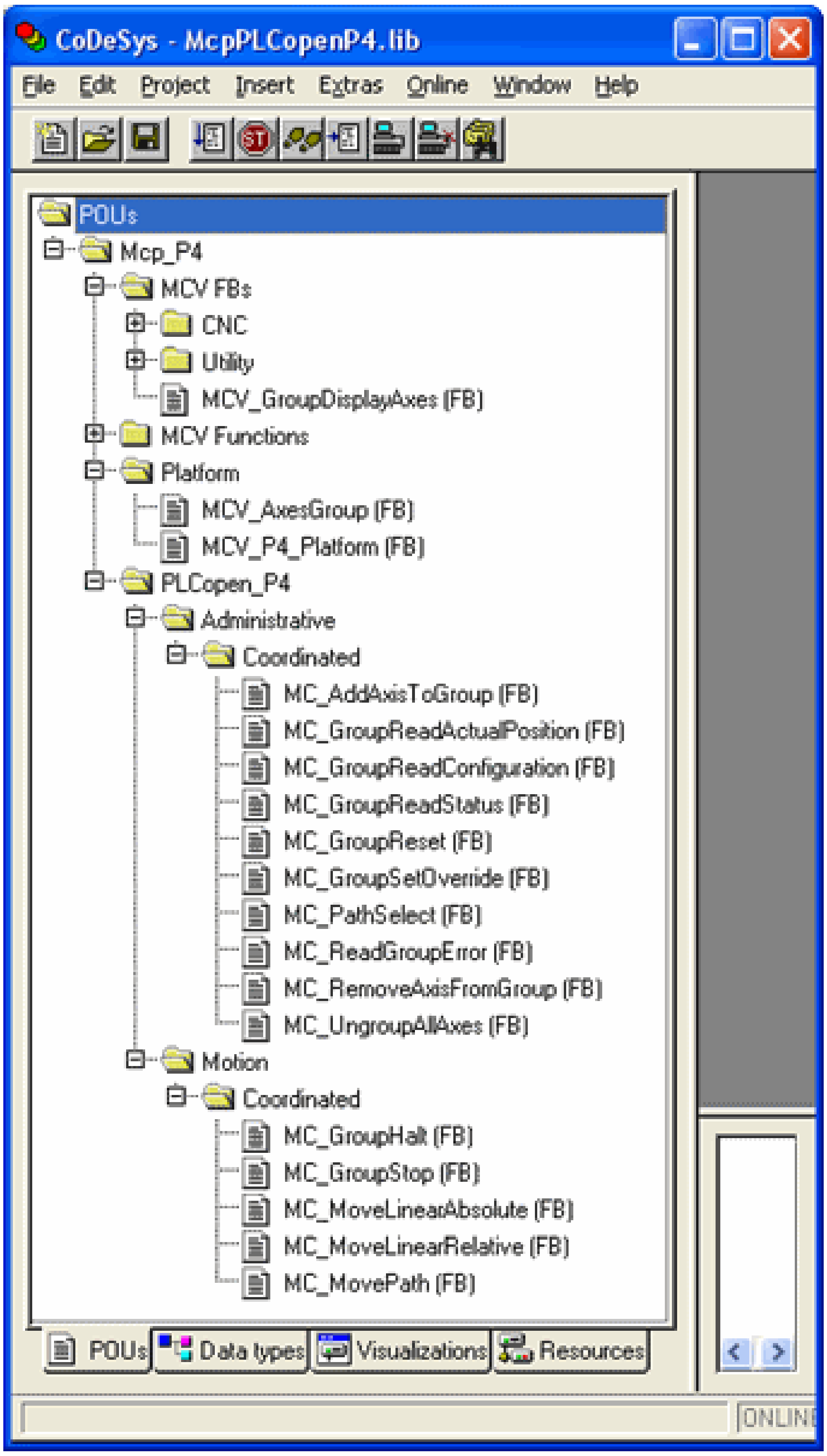
Fig. 6: Main program containing instance of FB MCV_P1_PLATFORM is invoked as first program of the task

The FB MCV_P1_PLATFORM assigns an AXIS_REF structure to the initialisation phase of each axis. This structure is then available as elements of the globally defined **g_array_axis_ref** array.

2.2.4 Axes group library – PLCopen Part4

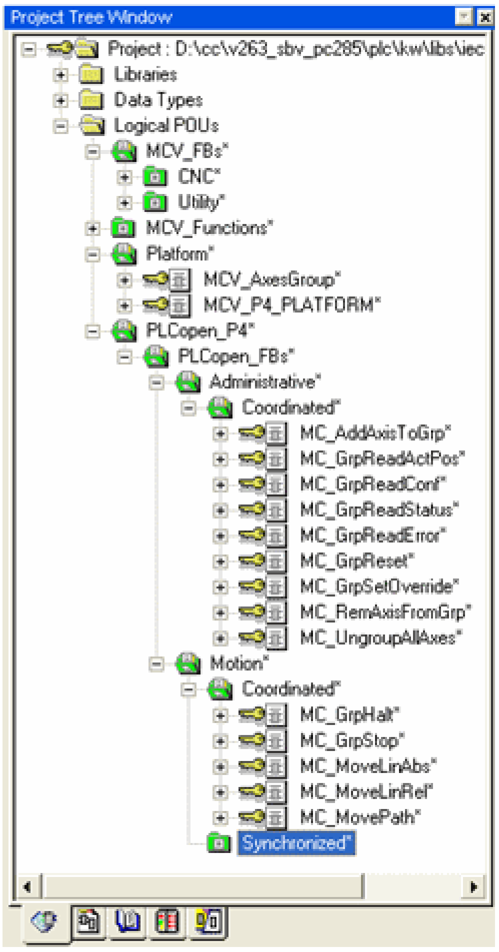
Besides FBs that comply with PLCopen specifications, the McpPLCopenP4.lib user library also defines FBs which cover additional functions and which must be used to implement an application. This library is referred to below as the Axis group library. The version scope may vary depending on the PLC platform used.

The figure below shows the structure of the motion library. The main elements in the library are then explained in greater detail.


CoDeSys - McpPLCopenP4.lib
File Edit Project Insert Extras Online Window Help
POUs
Data types
Visualizations
Resources
ONLINE

```
POUs
  Mcp_P4
    MCV FBs
      CNC
      Utility
      MCV_GroupDisplayAxes (FB)
    MCV Functions
    Platform
      MCV_AxesGroup (FB)
      MCV_P4_Platform (FB)
    PLCopen_P4
      Administrative
        Coordinated
          MC_AddAxisToGroup (FB)
          MC_GroupReadActualPosition (FB)
          MC_GroupReadConfiguration (FB)
          MC_GroupReadStatus (FB)
          MC_GroupReset (FB)
          MC_GroupSetOverride (FB)
          MC_PathSelect (FB)
          MC_ReadGroupError (FB)
          MC_RemoveAxisFromGroup (FB)
          MC_UngroupAllAxes (FB)
      Motion
        Coordinated
          MC_GroupHalt (FB)
          MC_GroupStop (FB)
          MC_MoveLinearAbsolute (FB)
          MC_MoveLinearRelative (FB)
          MC_MovePath (FB)
```

Fig. 7: Overview of the McpPLCopenP4.lib motion library in CoDeSys



```
Project Tree Window
Project : D:\cc\v263_sbv_pc285\plc\kw\libs\iec
  Libraries
  Data Types
  Logical POUs
    MCV_FBs*
      CNC*
      Utility*
    MCV_Functions*
    Platform*
      MCV_AxesGroup*
      MCV_P4_PLATFORM*
    PLCopen_P4*
      PLCopen_FBs*
        Administrative*
          Coordinated*
            MC_AddAxisToGrp*
            MC_GrpReadActPos*
            MC_GrpReadConf*
            MC_GrpReadStatus*
            MC_GrpReadError*
            MC_GrpReset*
            MC_GrpSetOverride*
            MC_RemAxisFromGrp*
            MC_UngroupAllAxes*
        Motion*
          Coordinated*
            MC_GrpHalt*
            MC_GrpStop*
            MC_MoveLinAbs*
            MC_MoveLinRel*
            MC_MovePath*
          Synchronized*
```

Fig. 8: Structure of McpPLCopenP4.zwt motion library

2.2.4.1 Function block MCV_AxesGroup

FB MCV_AxesGroup updates data of an AXES_GROUP_REF structure and has an AXES_GROUP_REF structure as input/output variable. This FB also performs the following tasks:

- Logging an Axis group to the MCE via the HLI. This is done by setting the "plc_present_w" flag in the channel-specific HLI area
- A check is made on initialisation if axes are assigned to an axis group. If this is the case, these axes are added to the PLC-internal axis group mapping without needing to command an MC_AddAxisToGroup FB.
- Adoption of the error messages reported by the MCE for each channel

In every PLC application which uses the PLCopen Part4 FB of the ISG-MCP, an instance of this FB must be created for every axis group used. This instance must be assigned an AXES_GROUP_REF structure in the form of **gAxesGroupRef[i]** as VAR_IN_OUT parameter.

To ensure this requirement, ISG-MCP contains the FB MCV_P4_PLATFORM [▸ 18]. This FB must called in a program of a PLC application. This ensures that the working data of one axis is updated in every PLC cycle.

2.2.4.2 Function block MCV_P4_PLATFORM

Notice

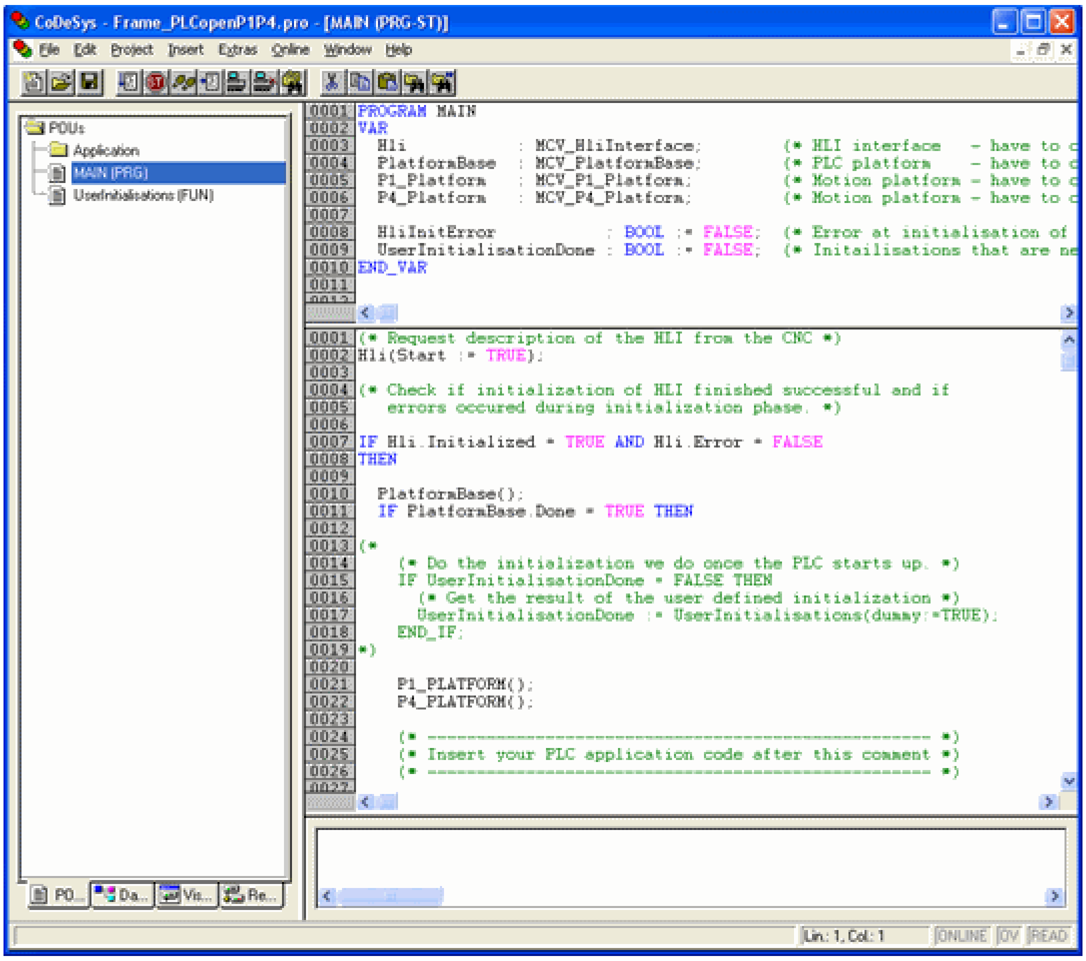
Definition for the MCP

a) The generic "MCV_AxesGroup" of the PLCopen axis groups are instanced in the ISG-MCP and implemented in the MCV_P4_PLATFORM FB.

b) Each PLC application that releases motion blocks using function blocks complying with PLCopen specification Part 4 must calculate exactly one instance of function block MCV_P4_PLATFORM before calculating other function blocks to release the motion task.

c) Application programmers must ensure that all PLCopen FBs used to program the application (e.g. motion sequence) are instanced and invoked in a single application program.

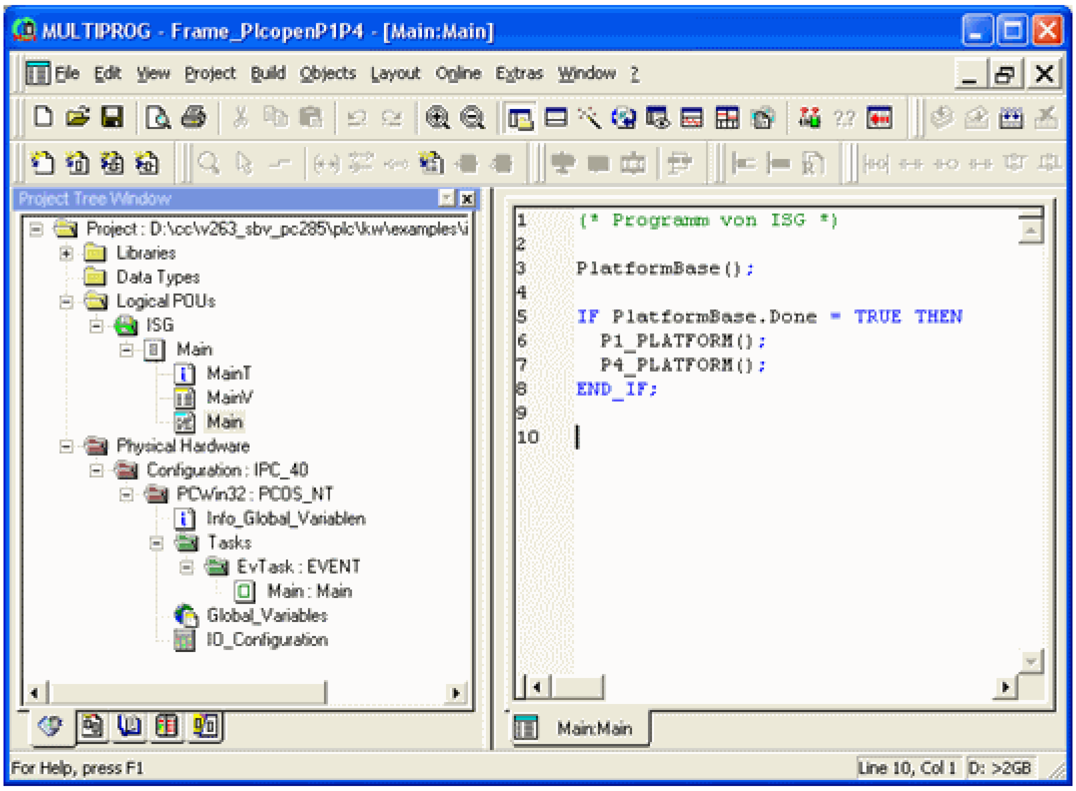
d) Before the first invocation of the MCV_P4_PLATFORM instance, the HLI (interface to the MC) must be initialised and the instance of the FB MCV_PlatformBase must notify the MCP of successful initialisation.


CoDeSys - Frame_PLCopenP1P4.pro - [MAIN (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
Application
MAIN (PRG)
UserInitialisations (FUN)
PO... Da... Vis... Re...
Lin.: 1, Col.: 1
ONLINE OV READ

```
PROGRAM MAIN
VAR
  Hli                      : MCV_HliInterface;        (* HLI interface   - have to c
  PlatformBase             : MCV_PlatformBase;        (* PLC platform    - have to c
  P1_Platform              : MCV_P1_Platform;         (* Motion platform - have to c
  P4_Platform              : MCV_P4_Platform;         (* Motion platform - have to c

  HliInitError             : BOOL := FALSE;   (* Error at initialisation of
  UserInitialisationDone   : BOOL := FALSE;   (* Initailisations that are ne
END_VAR
```

```
(* Request description of the HLI from the CNC *)
Hli(Start := TRUE);

(* Check if initialization of HLI finished successful and if
   errors occured during initialization phase. *)

IF Hli.Initialized = TRUE AND Hli.Error = FALSE
THEN

  PlatformBase();
  IF PlatformBase.Done = TRUE THEN

(*
    (* Do the initialization we do once the PLC starts up. *)
    IF UserInitialisationDone = FALSE THEN
      (* Get the result of the user defined initialization *)
      UserInitialisationDone := UserInitialisations(dummy:=TRUE);
    END_IF;
*)

    P1_PLATFORM();
    P4_PLATFORM();

    (* ------------------------------------------------------ *)
    (* Insert your PLC application code after this comment *)
    (* ------------------------------------------------------ *)
```

Fig. 9: PLC application frame for axis group applications in CoDeSys environment



```
(* Programm von ISG *)

PlatformBase();

IF PlatformBase.Done = TRUE THEN
  P1_PLATFORM();
  P4_PLATFORM();
END_IF;

```

Fig. 10: Main program with instance of FB MCV_P4_PLATFORM in Multiprog development environment

In the MCV_P4_PLATFORM initialization an AXES_GROUP_REF structure is assigned to each axes group, available as elements of the globally defined **gAxesGroupRef** array.

2.2.4.3 PLCopen function blocks

PLCopen specification Part4 divides the defined FB according to their usage into administrative or motion FB.

In these two areas, a distinction is made whether an FB only refers to the axis group (coordinated) or whether the FB commands a function which interacts with components outside the axis group (synchronised).

Following table shows the FB defined within PLCopen specification Part4 and is organized in the same way.

Notice

FBs marked by an * are not implemented in the Motion Library Part4. However, there may be FBs in the libraries with a similar functionality but specified by the control unit supplier.

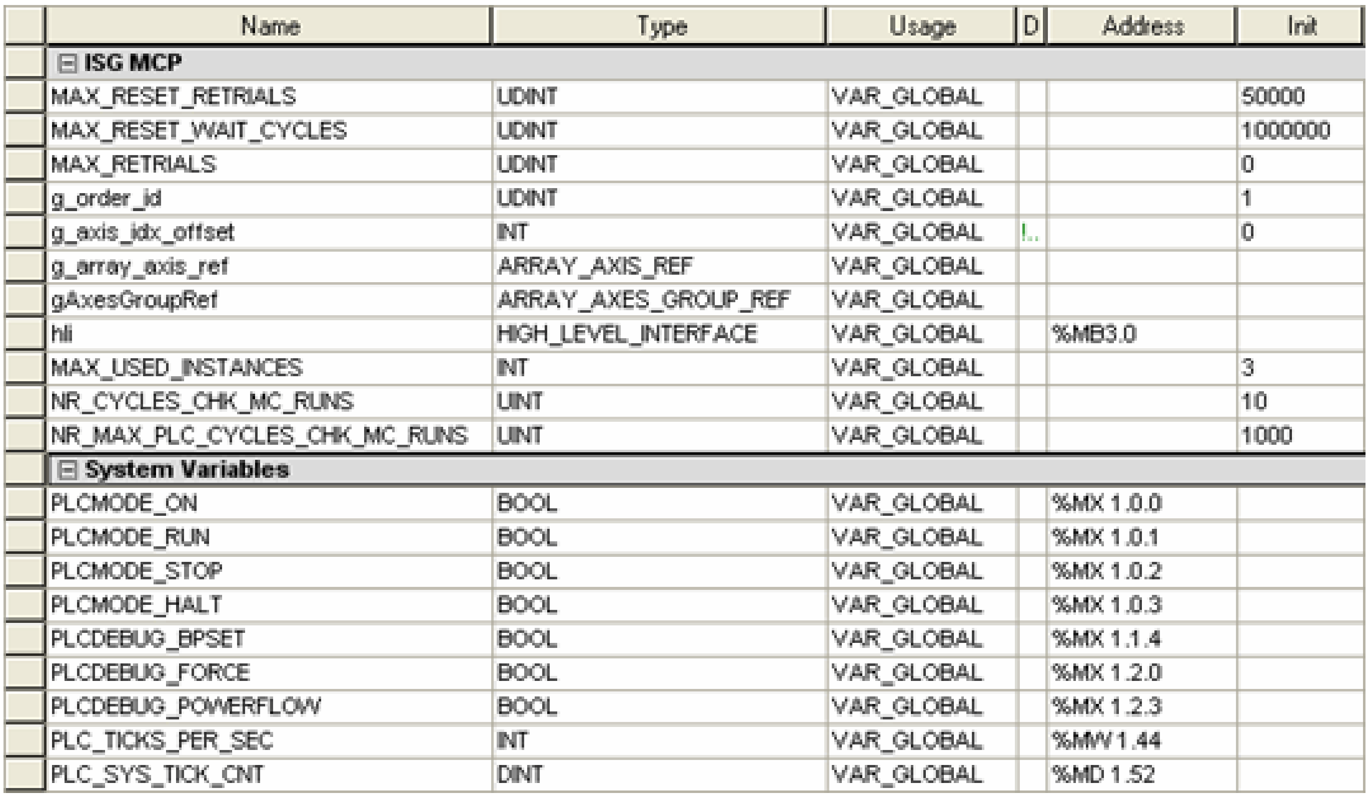
Classification of PLCopen-FB Part4 into administrative and motion-related FBs

Administrative	**Motion**	
Coordinated	**Coordinated**	**Synchronised**
MC_AddAxisToGroup	MC_GroupHome*	MC_SyncAxisToGroup*
MC_RemoveAxisFromGroup	MC_GroupStop	MC_SyncGroupToAxis*
MC_UngroupAllAxes	MC_GroupHalt	MC_TrackConveyorBelt*
MC_GroupReadConfiguration	MC_GroupInterrupt*	MC_TrackRotaryTable*
MC_GroupEnable*	MC_GroupContinue*	
MC_GroupDisable	MC_MoveLinearAbsolute	
MC_SetKinTransform*	MC_MoveLinearRelative	
MC_SetCartesianTransform*	MC_MoveCircularAbsolute*	
MC_SetCoordinateTransform*	MC_MoveCircularRelative*	
MC_ReadKinTransform*	MC_MoveDirectAbsolute*	
MC_ReadCartesianTransform*	MC_MoveDirectRelative*	
MC_ReadCoordinateTransform*	MC_MovePath	
MC_GroupSetPosition*		
MC_GroupReadActualPosition		
MC_GroupReadActualVelocity*		
MC_GroupReadActualAcceleration*		
MC_GroupReadStatus		
MC_GroupReadError		
MC_GrpReset		
MC_PathSelect		
MC_GroupSetOverride		
MC_SetDynCoordTransform*		

2.2.5 Global variables

Depending on the PLC development environment, it may be necessary to define global data in the PLC application that are used in user libraries.

In order to use the ISG-MCP, the global variables listed in the ISG MCP group must be defined for a resource:



Name	Type	Usage	D	Address	Init
⊟ **ISG MCP**					
MAX_RESET_RETRIALS	UDINT	VAR_GLOBAL			50000
MAX_RESET_WAIT_CYCLES	UDINT	VAR_GLOBAL			1000000
MAX_RETRIALS	UDINT	VAR_GLOBAL			0
g_order_id	UDINT	VAR_GLOBAL			1
g_axis_idx_offset	INT	VAR_GLOBAL	!..		0
g_array_axis_ref	ARRAY_AXIS_REF	VAR_GLOBAL			
gAxesGroupRef	ARRAY_AXES_GROUP_REF	VAR_GLOBAL			
hli	HIGH_LEVEL_INTERFACE	VAR_GLOBAL		%MB3.0	
MAX_USED_INSTANCES	INT	VAR_GLOBAL			3
NR_CYCLES_CHK_MC_RUNS	UINT	VAR_GLOBAL			10
NR_MAX_PLC_CYCLES_CHK_MC_RUNS	UINT	VAR_GLOBAL			1000
⊟ **System Variables**					
PLCMODE_ON	BOOL	VAR_GLOBAL		%MX 1.0.0	
PLCMODE_RUN	BOOL	VAR_GLOBAL		%MX 1.0.1	
PLCMODE_STOP	BOOL	VAR_GLOBAL		%MX 1.0.2	
PLCMODE_HALT	BOOL	VAR_GLOBAL		%MX 1.0.3	
PLCDEBUG_BPSET	BOOL	VAR_GLOBAL		%MX 1.1.4	
PLCDEBUG_FORCE	BOOL	VAR_GLOBAL		%MX 1.2.0	
PLCDEBUG_POWERFLOW	BOOL	VAR_GLOBAL		%MX 1.2.3	
PLC_TICKS_PER_SEC	INT	VAR_GLOBAL		%MW 1.44	
PLC_SYS_TICK_CNT	DINT	VAR_GLOBAL		%MD 1.52	

Fig. 11: Global variables required for use of the ISG-MCP in the Multiprog development environment

2.3 Safety concept and compliance with EN775

2.3.1 General details of the software safety concept

Software safety functions in the above sense are fundamentally realised in the ISG-MCE or the MCP.

The safe state is always the default state, i.e. this default state can only be *deactivated* by means of special safety-relevant function blocks intended for this purpose.

The ISG delivery **does not** include the function blocks required for deactivation.

As the HLI is a memory-coupled interface, a realisation has been chosen for communication of safety-relevant commands that ensures that the one-time deactivation of a safety-relevant function ordered by setting a memory position does not continue to apply if the PLC no longer calls up the applicable safety function block.

2.3.2 Actual speed monitoring

Actual speed monitoring is realised in the ISG-MCE.

For linear axes, the limit for speed monitoring is compiled at a fixed value of 250 mm/s.

For rotary axes or spindles, parameter definition must be possible on the basis of component diameter. This is why the limit is set in the ACHS_MDS list using the gear parameter “vb_monitor” (P-AXIS-00311). If the parameter is not in the list or is "0", the actual speed is monitored for"vb_not_referenced" (P-AXIS-00268).

- In the SAI the error message 60241 (P-ERR-60241) "Programmed speed exceeds monitoring limit” is output if a motion task is commanded at a higher speed.
- In the ISG-MCE, speed monitoring is active by default and can be deactivated with a special safety function, which must be called up cyclically.
- The actual speed is monitored cyclically in the system clock in the BF LR. If the permissible actual speed is exceeded, the error message 70225 P-ERR-70225 "Max. actual speed exceeded during active speed monitoring" is issued and the axis is stopped by closing of the brake and removal of controller enabling.

2.3.3 Bidirectional congruence checking of the HLI memory interface

The ISG FBs send commands to the ISG-MCE via the so-called HLI. As this is a memory-coupled interface, congruent post-definition of this interface in the PLC is the prerequisite for correct functioning of the overall system. This is why a congruence check is realised in the MCP library to ensure that the HLI interface is not operated with an incorrect HLI emulation.

2.3.4 Priority of the MC_Stop FB

The HLI of the ISG-MCE has, **per axis, precisely one** commanding and acknowledgement interface for MC commands. The commanding and acknowledgement interface are each realised as a consumption data item. Accordingly, a maximum of one FB command can be issued in each PLC cycle. If a second FB is triggered in one PLC cycle, it cannot be asserted. If (within one PLC cycle) the next command is MC_Stop, the one already existing in the command memory is overwritten by this MC_Stop, and so MC_Stop always has the highest priority.

2.3.5 Speed monitoring during active torque limiting

An additional monitoring function is realised for actual speed monitoring during active torque limiting. This can be parameterised in the ACHS_MDS parameter “vb_torq_limit_max“ (P-AXIS-00314).

- The actual speed during active torque limiting is also monitored cyclically in the BF LR. If the permissible actual speed is exceeded, the error message "Max. actual speed exceeded during active torque monitoring" (P-ERR-70220) is issued and the axis is stopped by closing of the brake and removal of controller enabling.

2.3.6 Preventing unintentional MCFB motions in the T1 mode

Setting up modes:

- T1 mode = setting up with reduced speed
- T2 mode = setting up without reduced speed

The ISG-MCE has no direct knowledge of the system's mode. The PLC programmer, to whom this information is available, can leave the safe state (= standstill) with the safety function blocks. The following requirement applied to compliance with EN775:

In critical state, i.e. an operator is located in the machine hazard zone and mode T1 or T2 is active, it should only be possible to command motions by means of conscious operator actions (in the case of a robot: confirmation key + START key).

In this state, release of the start key must lead only to feed hold, but not to command aborting.

The drive's enabling signals generally disappear anyway when the mode is changed from automatic to T1 or T2.

When the confirmation key is pressed, PLCopen commands are buffered, but the motion does not yet start.

The PLC programmer can then use pressing of the start key to supply the safety function block, thus ensuring the customary functionality of the start key.

2.3.7 Safety low speed for non-referenced axes

If an axis is not yet referenced/adjusted or has lost the reference point/adjustment, the dimension reference to a fixed point on the machine has also been lost and so software limit switch monitoring cannot take place.

Therefore, in this state the axes cannot be absolutely positioned. Only MC_MoveRelative and MC_MoveVelocity are possible. These relative positioning methods are run at reduced speed, which can be configured in the ACHS_MDS parameter

```
getriebe[0].vb_not_referenced
```

can be configured (P-AXIS-00268).

2.4 Details of realisation within the ISG-MCP

The aspects of the ISG-MCP realisation will be discussed in this section in the case of which the behaviour of the FB cannot be precisely forecast with the PLCopen specification 1.0 alone.

Reasons for this may be:

- Insufficient definitions in the PLCopen specification 1.0
- Characteristics of the motion control engine

2.4.1 Startup

Various initialisation processes of the ISG-MCP run through in the event of a cold start of the PLC program.

During this initialisation phase, AXSD is in the state 0 (INIT_STATE).

During this phase, all FBs of the MCP behave passively, i.e. they do not have write access to the HLI. If an FB sends a command before the initialisation phase has ended, e.g. as a result of initialisation of an "Enable" input with "TRUE", the FB reports the error ERR_PO_AX_TNA_INIT_STATE (P-ERR-44013).

2.4.2 General details of the operating principle of the PLCopen FBs

Internally, the PLCopen FBs do not possess any interpolation functionality. Instead, this merely serve to send motion commands to the motion control engine and to take receipt of relevant acknowledgements.

2.4.3 Realisation of the FBs

The PLCopen FBs internally map the state diagram below

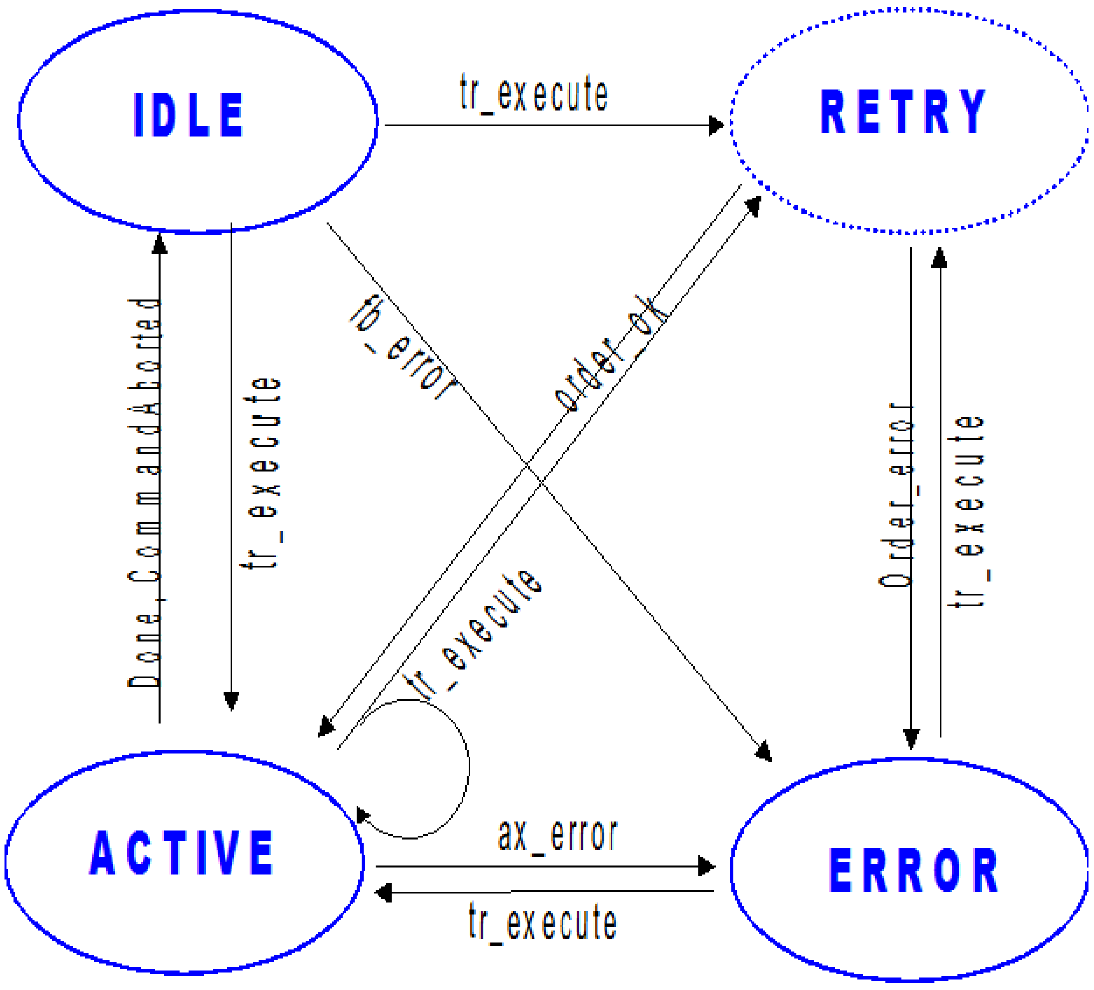


Fig. 12: States in an FB.

The state machine in IEC 61131-3 Structured Text displayed below shows the framework for implementing the commands in an FB: The individual actions are written in the pseudo code.

```
(*===========================================================*)
(*      State distributor for commanding the HLI
(*===========================================================*)
CASE fb_state OF
(*===========================================================*)
(*===========================================================*)
FB_IDLE,
FB_ERROR:(*                                                  *)
IF ( tr_execute.Q = TRUE OR retry ) THEN
  (* checking of the FB's input parameters                   *)
  (* check whether transition is allowed                     *)
  (* try to send the MC order.                               *)
  (* IF (sendorder = OK) THEN fb_state := FB_ACTIVE; END_IF *)
END_IF


(*===========================================================*)
(*===========================================================*)
ACTIVE:(*                                                    *)
IF ( tr_execute.Q = TRUE OR retry) THEN
  (* check whether FB's ax_ref connection has changed since idle state*)
  (* checking of the FB's input parameters *)
  (* check whether transition is allowed *)
  (* try to send the MC order. *)
END_IF
(* collection of Acknowledge *)
(* IF (Acknowledge = OK) THEN fb_state := FB_IDLE; END_IF *)

(*===========================================================*)
(*===========================================================*)
ELSE
(*// default: Impermissible state *)
END_CASE;
```

The FBs are not aware of any reset transition and, instead, after a preceding FB_ERROR that follows retriggering of the FBs, an attempt is simply made to assert the new command issued. This is why the state distributor does not differentiate between FB_IDLE and FB_ERROR with respect to HLI commands issued.

The FB_RETRY state does not appear as an explicit state in the state distributor, but is kept in a variable in the states FB_IDLE,FB_ERROR and FB_ACTIVE in case the FB in the corresponding state cannot assert commands issued at first go.

The unique command number is kept as a global PLC data item and incremented by the FBs only at the time **after successful commands issued from the FB_IDLE and FB_ERROR states**, and is noted in its instance data.

When a command is issued out of the FB_ACTIVE state, a new command with the same command number is sent to the MC and an internal counter in the FB, which counts how many commands pertaining to one command number, is incremented.

This method dispenses with complex administration of the command numbers that have been issued and the associated acknowledgements received.

When an FB is triggered, it checks whether the state transition that **would be** triggered by execution of the FB in the FBSD is permitted at all in the momentary state of the FBSD ("transition allowed"). If this is not the case, the command is not sent at all and, instead, the FB reports a command issuing error (= error at FB level). In such a case, the axis state never changes because no command at all has been sent. See the next section for more details.

2.4.4 Interaction of the FB with the FBSD and error handling

The "FB state behaviour" state engine described in the PLCopen specification always refers to one axis. This is why it makes sense to keep this axis state in the PLC's axis-specific working data and to make it available to all FBs via the AXIS_REF structure. The index of the corresponding Ax-HLI interface is also noted in the AXIS_REF structure.

As the axis state is kept in the axis-specific working data of the PLC, it is obvious that this FBSD can initially only contain the command issuing state.

This is also expressed in the following PLCopen rule:

```
The axis is always in one of the defined states (see diagram below). Any
motion job is a transition that changes the state of the axis …
```

When an FB receives a command, it must check the command's admissibility on the basis of the FBSD's current state.

2.4.4.1 Error handling at the FBSD level

At this point you may come to the conclusion that the FB places the FBSD in an error state in the event of an inadmissible command because it is a command issuing state automaton. According to the PLCopen specification, however, this is **not** the case because the following applies to the FBSD:

```
Note 3: The transition Error refers to errors from the axis and axis
control and not from the Function Block instances.
These axis errors may also be reflected in the output of the Function
Blocks 'FB instances errors'.
```

This means that a motion FB never places the FBSD in the ERROR state, but is only an error that is reported by the motion controller. To this end, it is necessary for an axis-specific handler process to obtain the error messages from the HLI and to file them in the axis-specific work data, i.e. in the FBSD. The individual FBs then see the error state of the axis via their AXIS_REF.

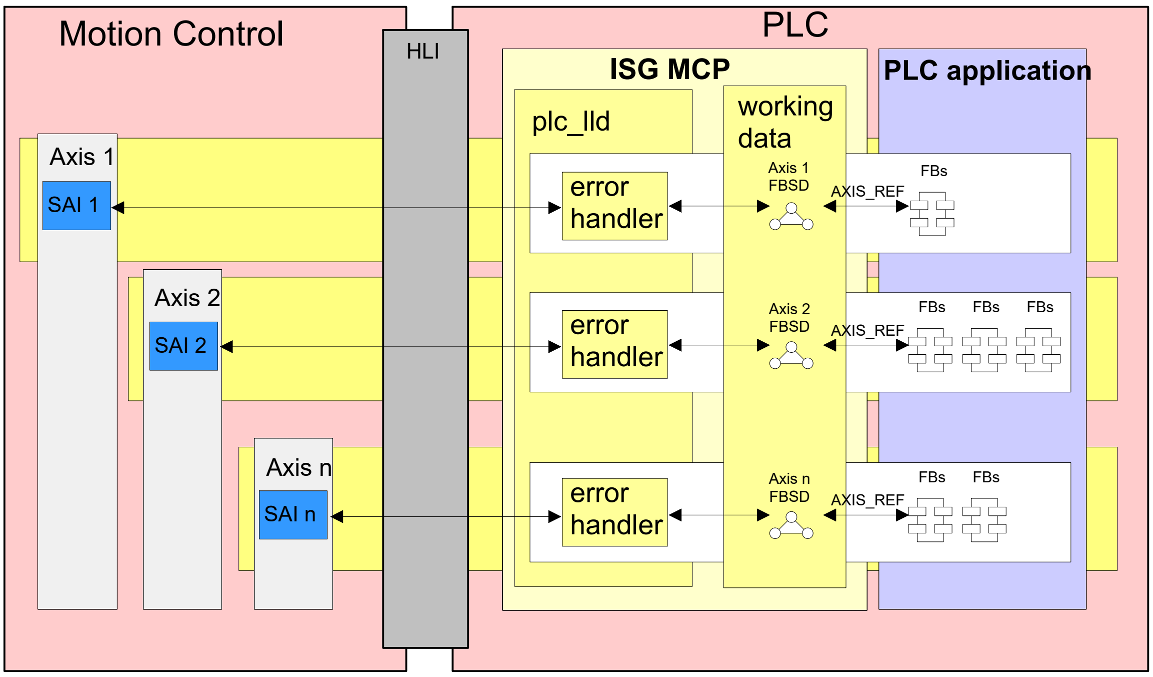


Fig. 13: Error handler supplies axis-specific FBSD work data

2.4.4.2 Error handling at the FB level

The individual FBs' error outputs refer to errors that have occurred within the scope of command issuing to an FB instance, not to the errors of the axis. This is why incorrect command issuing sets only the commanded FB to an error state. The incorrect command itself is not even issued by the FB and so the axis (FBSD) is not even set to an error state. Therefore, other FB instances located on the same axis do not notice anything.

2.4.4.3 Defined errors at the FB level

The specific errors that may occur in individual FBs are described in the diagnostic manual (DIAG).

2.4.4.4 Axis errors from the motion controller

The motion controller uses the axis-specific interfaces of the HLI to provide messages about an axis for the PLC. The information is exchanged via a data structure of the type HLI_ERROR_SATZ.

```
TYPE
  HLI_ERROR_SATZ :
  STRUCT
    error_id         :  UDINT;   (*Error number*)
 (*System time when error occurs*)
fb_zeitangabe: HLI_FB_ZEITANGABE;
    bf_type          : WORD;     (*BF type*)
behebungs_klasse: WORD;     (*Error rectification class*)
reaktions_klasse: WORD;     (*Error response class*)
    reserved         : WORD;
  END_STRUCT;
END_TYPE
```

Instances of the FB MCV_Axis check this axis-specific area in every PLC cycle because these are instanced in the MCV_P1_PLATFORM program and must be integrated first in the PLC task in conformity with the explanations in Section 1.2.3.3 [▸ 13] of this program description. The MCV_Axis extracts **every** newly occurring message and transfers it to the AXIS_REF structure of the assigned axis, which contains one field for six data structures of the type HLI_ERROR_SATZ.

If a message has been classified as an error, the MCV_Axis instance sets the current state of the axis state diagram (AXSD) to **ERROR_STOP**.´

Notice

Error messages are all those messages for which the value of the variable is **behebungs_klasse > 0**.

If the value of behebungs_klasse = 0, the message is a warning.

The **ERROR_STOP** state is detected by the other PLCopen FB instances to which the same axis is assigned. Consequently, they set their "Error“ output variable to TRUE and the value **1** (ERR_PLC_AX_MC, see P-ERR-40001) is indicated at the “ErrorID” output variable.

2.4.5 Versioning

Corresponding to the general conditions given by the PLC run time system and the requirement for availability even without a development system or a running PLC application, the version information is stored on the one hand in the names of the PLC libraries and, on the other hand, function blocks are implemented that check, in the application, the versions of the individual components (PLC libraries and HLI definitions at the MC or PLC end) of the ISG-MCP.

2.4.5.1 Version check by FBs

The implementation of version monitoring by FB is based on the principle that each component of the ISG-MCP checks the version of the interfaces and PLC libraries it depends on itself. This is why some FB are implemented in the **HLI library** and in the **motion library**. These check the components or they return the version of the library

Generally, the user does not need to bother about this check because it is already done during startup of the PLC application by the instances of the MCV_Axis FB. These MCV_Axis instances are instanced in the MCV_P1_PLATFORM program which is invoked cyclically.

If inconsistencies occur, the other FB instances of the application are informed of them and they indicate FALSE at their "error" output and a specific error code at the the "error_id" output

2.4.6 Further general system characteristics

- Limit switch monitoring:(not effective with modulo axes): when a motion path limit is detected, braking takes place at the rapid traverse acceleration values (limit acceleration at the power limit, see P-AXIS-00004 or P-AXIS-00005, P-AXIS-00006), not at the default acceleration.
- The Discrete Motion state (MC_MoveRelative) aborts the Continuous Motion state at high speed, with the result that the deceleration distance amounts to more than one modulo range. The expected behaviour does not transpire from the PLCopen specification. In the case of modulo axes, the ISG-MCP behaves as follows: the target position is noted at the moment of command aborting. The axis is braked over so many revolutions to the position as are necessary to keep to the acceleration stipulations. In the case of linear axes, the entire braking distance is run backwards because there is only one mathematical possibility of reaching the target position.

3 Appendix 1: PLC application programming best practice

3.1 General

The PLCopen specifications define a specific behaviour of the PLCopen FBs. An exact knowledge of the PLCopen specifications is the basic prerequisite for successful PLC application programming using the PLCopen FB and the ISG-MCP. In addition, it is expected that programmers know how to handle the PLC programming system confidently for the applications below.

3.2 Essential details in brief

3.2.1 Behaviour of the PLCopen-FB´s "execute" and "done" inputs!outputs

A PLCopen FB evaluates only the **RISING EDGE** of the "execute" signal.

That is to say that, before an FB can be commanded again, it must be called up at least once with execute = FALSE!

The "done" signal of a PLCopen FB is deleted **only** on the basis of the **FALLING EDGE** of the "execute" signal.

That is to say, if the "done" output of an FB is connected to the "execute" input of a second FB, for example, the following problem may arise in connection with trigger commanding:

If the first FB is triggered and "execute" becomes FALSE before "done" becomes TRUE, this "done" remains at the first FB until its "execute" runs through another DOWN EDGE of the "execute" signal. As the second FB is linked directly to the first one, its "execute" cannot detect a rising edge again until the first FB has undergone complete triggering. Up to then, however, it is **blocked!**

3.2.2 The crux of the matter: command assertion and acknowledgement

The HLI of the ISG-MC has, **per axis, precisely one** commanding and acknowledgement interface for MC commands. The commanding and acknowledgement interface are each realised as a consumption data item. Accordingly, a maximum of one FB command can be issued in each PLC cycle.

If a second FB is triggered in one PLC cycle, it returns the FB-specific error: **4** FB_ERR_MC_DID_NOT_TAKE_ORDER because the commanding interface is blocked by the previous command.

This is why the following rule applies:

No more than one PLCopen FB command may be issued per axis and PLC cycle.

Although a maximum of one command may be issued per axis and PLC cycle, from the PLC's point of view there may be several commands "en route", for which acknowledgements are expected.

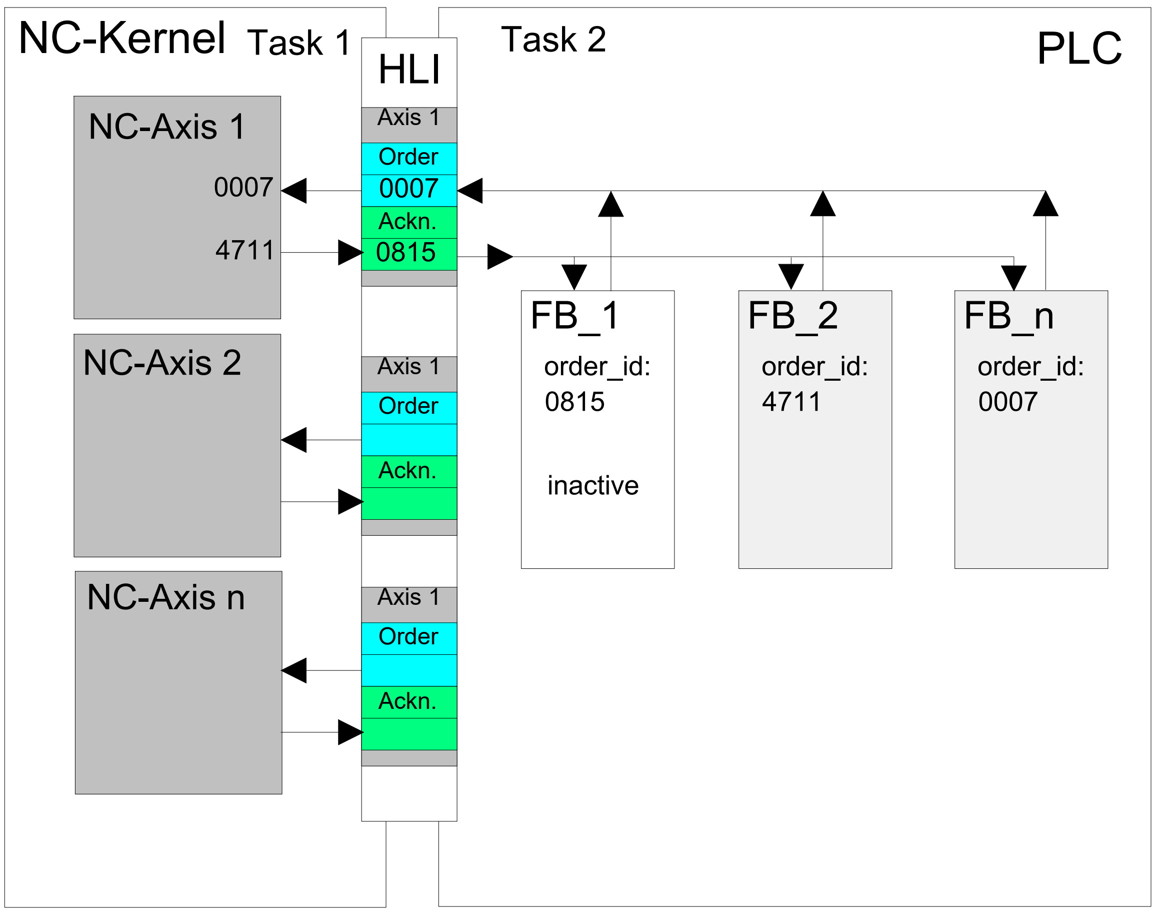


Fig. 14: State of congestion if an FB is no longer called up

Each FB instance extracts only those acknowledgements (command numbers) that is has commanded. A state of congestion arises if an FB, after having sent a command, is no longer called up before receiving its acknowledgement. The other, still active ones, can then no longer receive an acknowledgment.

Such a situation can only arise due to an error in the PLC application program or as a result of undue shutdown and startup of the PLC while the NC is running.

This is why the following further important rule applies:

After commanding, a PLCopen FB must be called up until it sets one of its" done"," command_aborted" or" error" outputs.

To detect such a problem, a monitoring function can be integrated in the PLC program to check whether one and the same acknowledgement is applied to the HLI for more than two PLC cycles. This cannot be the case if the PLC program is error-free.

3.3 PLC programming tips and tricks

PLC application programming is typically done in step sequences. An unexpected behaviour of the application program may arise if the aforementioned commanding and call types are mixed.

With regard to the **commanding method**, a PLCopen FB can be distinguished according to:

- Trigger commanding ("Execute" TRUE only for one cycle)
- Level commanding ("execute" is applies at least until" done" = TRUE)

PLC programmers should give thought beforehand to which of the two commanding methods they wish to use and should stick to it within one project.

With regard to the **call method**, a PLCopen FB can be distinguished according to:

- Call within all FBs in each PLC cycle
- Call only of the relevant FB in each step

Here also, PLC programmers should give thought beforehand to which of the two call methods they wish to use and should stick to it within one project.

3.4 Tips on runtime optimisation

In large applications with multiple axes, it may be necessary to invoke function block instances time-optimised, i.e. the function block instances are only executed when they are required.

The following short code section in StructuredText illustrates the technique for reducing the runtime of an application. The function block “MC_MoveVelocity” is used as an example here.

The variable “MC_MoveVelocity_Active” controls the function block. It may assume the following values:

Value	Meaning
0	Function block is not executed.
1	Function block is executed; input “Execute”/”Enable” is TRUE.
2	Function block is executed until the requested acknowledgement is set (e.g. “Done”, “CommandAborted” etc.). “Execute“/”Enable“ is FALSE.

At the beginning, “MC_MoveVelocity_Active” is set to 1 depending on the input “Execute”/”Enable”. The function block “MC_MoveVelocity” is executed in this setting. This state remains until the input is reset.

```
IF ( Execute_MoveVelocity   = TRUE ) AND
   ( MC_MoveVelocity_Active = 0 )     THEN
  MC_MoveVelocity_Active := 1;
END_IF;
```

Only when the input “Execute”/”Enable” is set to FALSE does the value of “MC_MoveVelocity_Active” change to 2. The function block is then executed until an acknowledgement is applied to the output (here: “Done”, “CommandAborted”, etc.). In case of error, which is not considered here, proceed in a similar manner.

```
IF MC_MoveVelocity_Active > 0 THEN
  MC_MoveVelocity( Axis          := AxisReference,
                   Execute       := Execute_MoveVelocity,
                   Velocity      := Velocity_MoveVelocity,
                   Acceleration  := Acceleration_MoveVelocity,
                   Deceleration  := Deceleration_MoveVelocity,
                   Jerk          := Jerk_MoveVelocity,
                   Direction     := Direction_MoveVelocity );

  AxisReference                := MC_MoveVelocity_0.Axis;
  InVelocity_MoveVelocity      := MC_MoveVelocity_0.InVelocity;
  CommandAborted_MoveVelocity  := MC_MoveVelocity_0.CommandAborted;
  Error_MoveVelocity           := MC_MoveVelocity_0.Error;
  ErrorID_MoveVelocity         := MC_MoveVelocity_0.ErrorID;

  IF   MC_MoveVelocity_Active      = 2     AND
      (InVelocity_MoveVelocity     = TRUE   OR
       CommandAborted_MoveVelocity = TRUE ) THEN
    MC_MoveVelocity_Active := 0;
  ELSIF Execute_MoveVelocity = FALSE THEN
    MC_MoveVelocity_Active := 2;
  END_IF;
END_IF;
```

4 Appendix 2: HelloWorld with the ISG Motion Control Platform

4.1 "Multiprog" programming example

Assume that the task is for the first axis in the system to move relative to the previous position by specifying distances. In addition, the current position of the axis is to be displayed.

4.1.1 Step 1: Inserting the required libraries

To execute a motion task using PLCopen FB, the PLC libraries

hli_lib.mwt, memory simulation between the PLC and MC

McpBase.mwt set up a link to the MC, supply data structures and the FBs which are used in other libraries

McpPLCopenP1.mwt, FB according to PLCopen specifications Parts 1 and 2

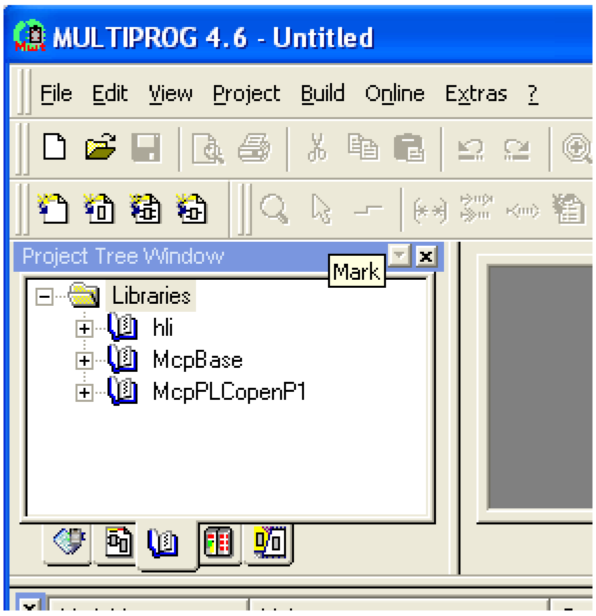
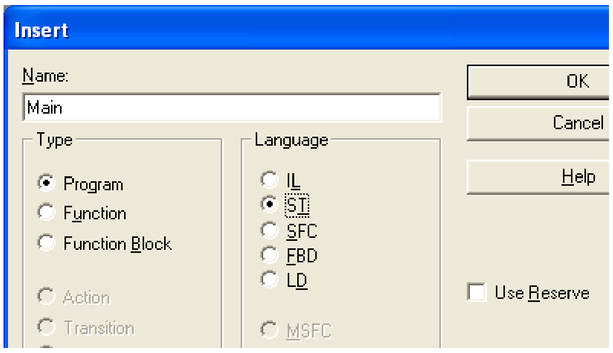


Fig. 15: Integrate required libraries in project

4.1.2 Step 2: Creating the PLC Main and HelloWorld programs

In this example the Main program is programmed in Structured Text (ST) and the HelloWorld is programmed as Function Block Diagram (FBD).


Insert

Name: Main

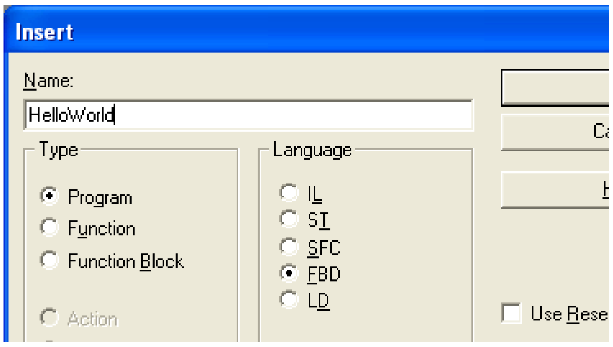
Type: [✓ Program] Function / Function Block / Action / Transition

Language: IL / [✓ ST] / SFC / FBD / LD / MSFC

OK / Cancel / Help

Use Reserve

Fig. 16: Create the Main program in ST


Insert

Name: HelloWorld

Type: [✓ Program] Function / Function Block / Action

Language: IL / ST / SFC / [✓ FBD] / LD

Use Rese

Fig. 17: Creating the HelloWorld program in FBD

After the programs are created, they appear in the project tree under “Logic POEs”.

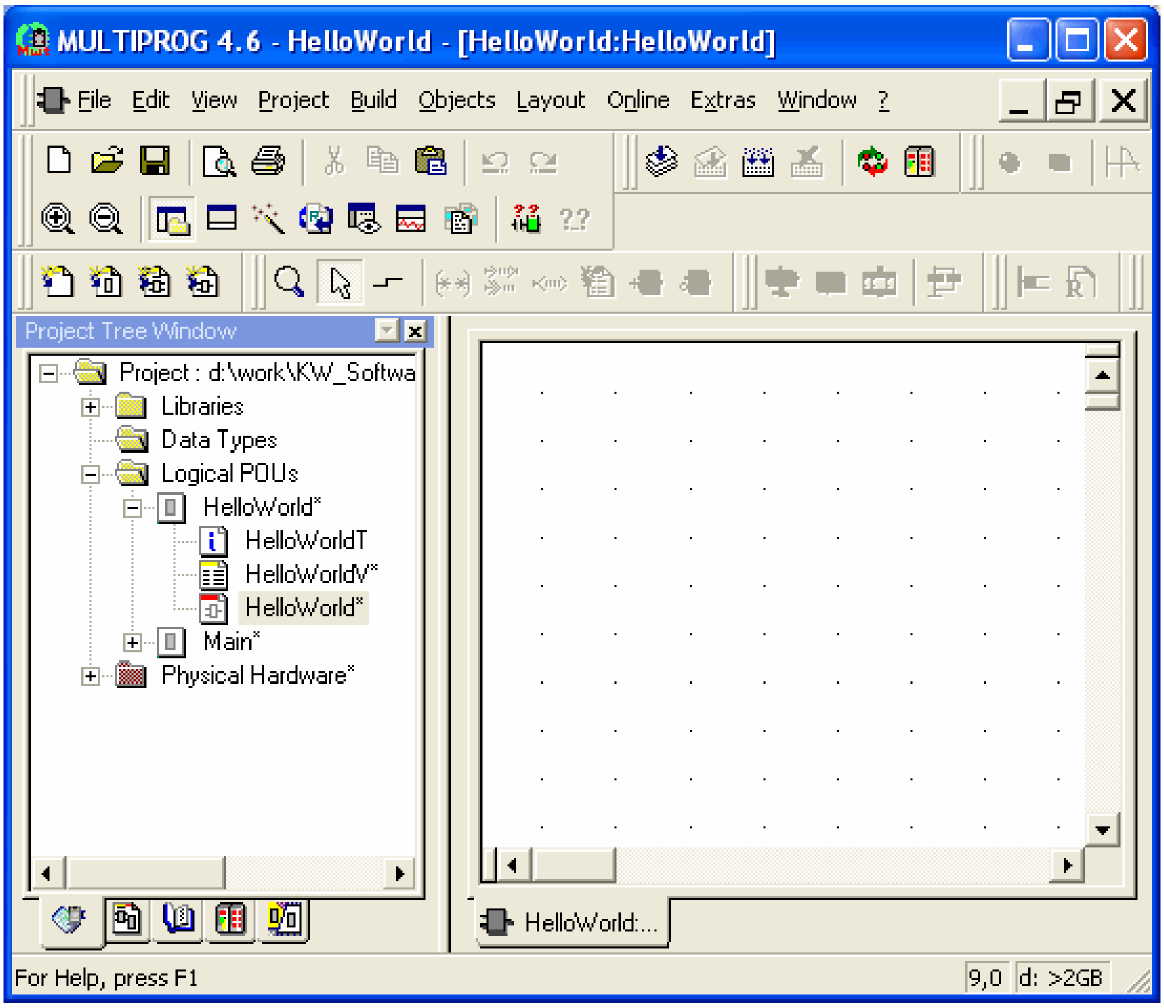


Fig. 18: Assigning the Main and HelloWorld programs in the project tree

4.1.3 Step 3: Implementing the Main program

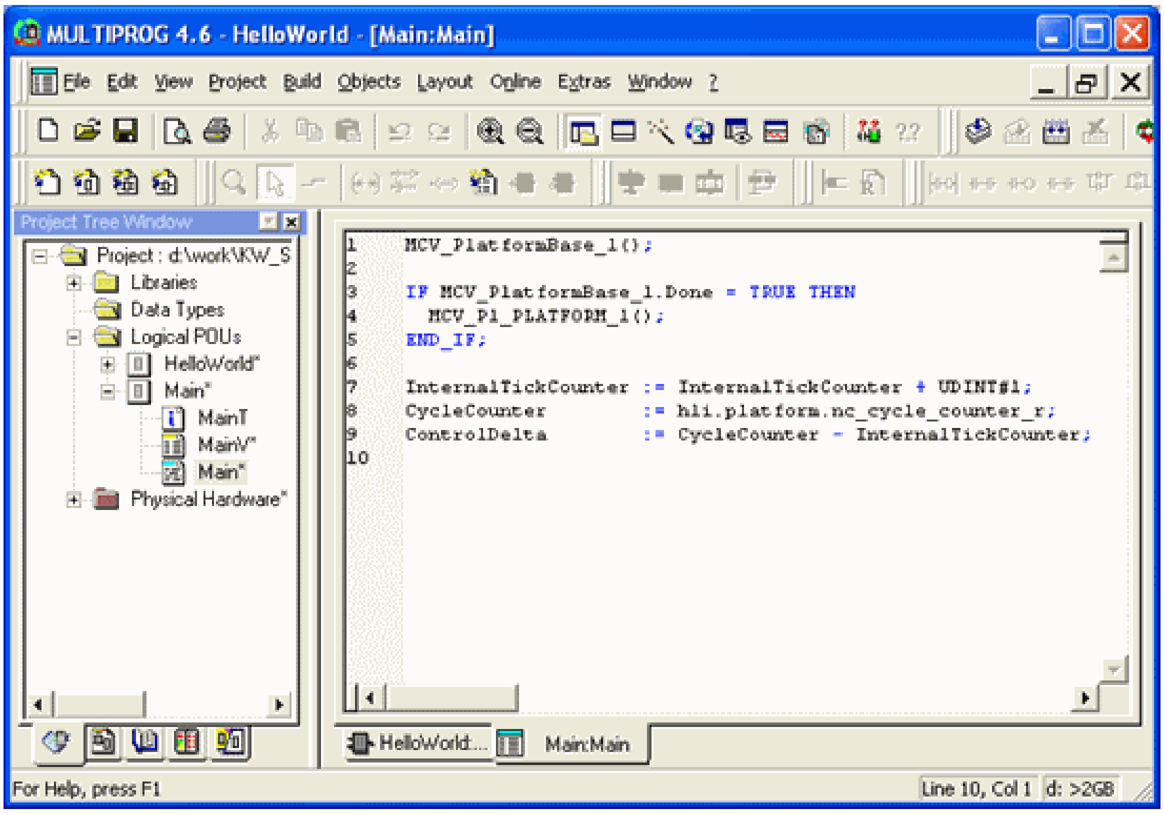
To execute motion tasks using FBs according to PLCopen specification Part1, an instance of the MCE platform FB **MCV_PlatformBase** and an instance of the Motion platform FB **MCV_P1_PLATFORM** must be invoked cyclically.

For this reason, an instance is created in the Main program of every FB, as listed in the table:

Instances of FBs used in the Main program

FB type	Instance name	Remarks
MCV_PlatformBase	MCV_PlatformBase_1	Sets up link to MC
MCV_P1_PLATFORM	MCV_P1_PLATFORM_1	Updates the axis references cyclically.

And the following code is implemented:in ST



```
MCV_PlatformBase_1();

IF MCV_PlatformBase_1.Done = TRUE THEN
  MCV_P1_PLATFORM_1();
END_IF;

InternalTickCounter := InternalTickCounter + UDINT#1;
CycleCounter        := hli.platform.nc_cycle_counter_r;
ControlDelta        := CycleCounter - InternalTickCounter;

```

Fig. 19: Implementing the Main program

4.1.4 Step 4: HelloWorld program: Instancing PLCopen FBs

The FB instances listed below are required to execute the task and are created in the HelloWorld program in which the motion task is to be implemented:

Instances of the FBs used in the HelloWorld program

PLCopen FB	Instance name	Remarks
MC_Power	MC_Power_1	Serves to set controller and feed enabling
MC_ReadActualPosition	MC_ReadActualPosition_1	Shows the position of the axis at the OUT variable "position"
MC_MoveRelative	MC_MoveRelative_1	Moves the axis by the value of the IN variable "distance" relative to the current position.

Display of FBs instanced in the HelloWorld:program

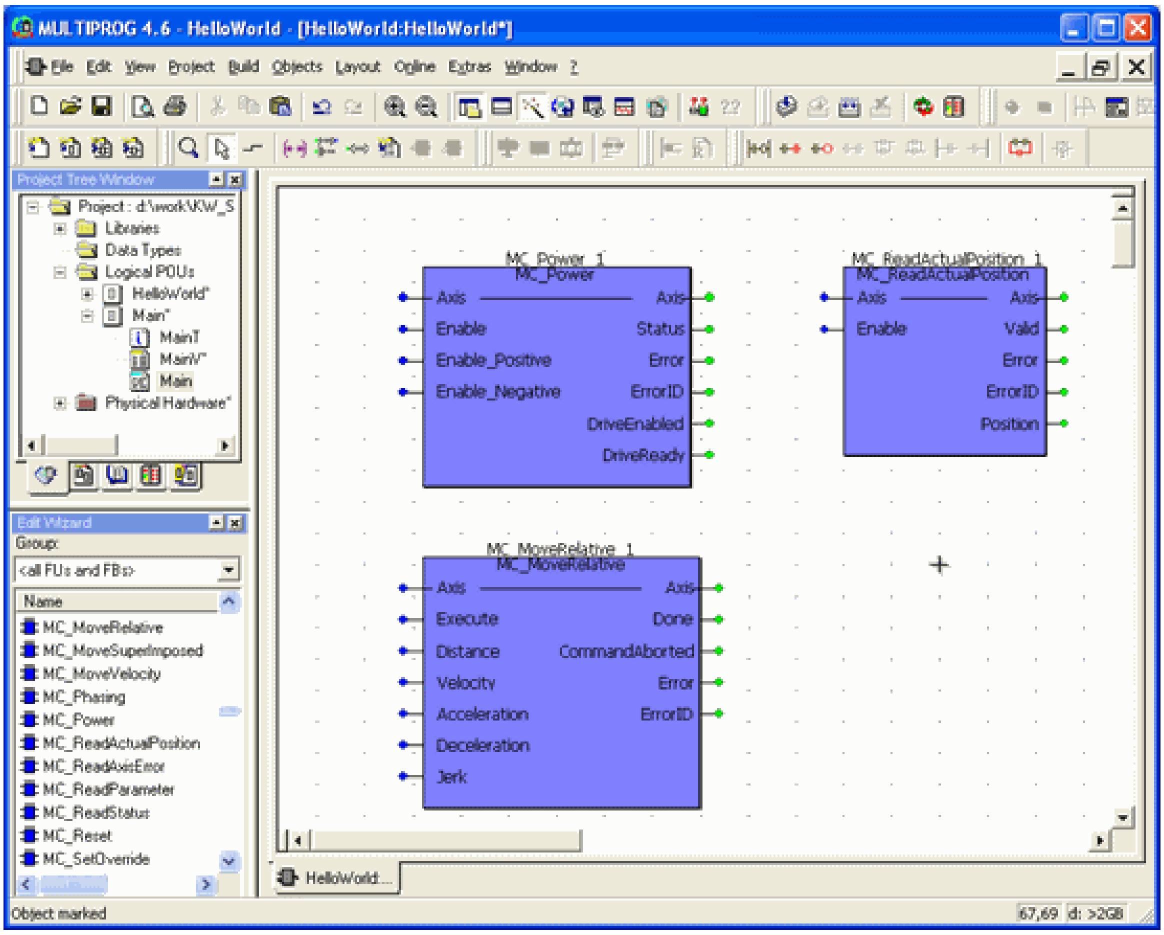


Fig. 20: Instances of PLCopen FBs in the HelloWorld world

4.1.5 Step 5: Linking the axes to PLCopen FBs

The axis reference g_array_axis_ref[0] which links to the first axis in the system is applied to all PLCopen FBs..

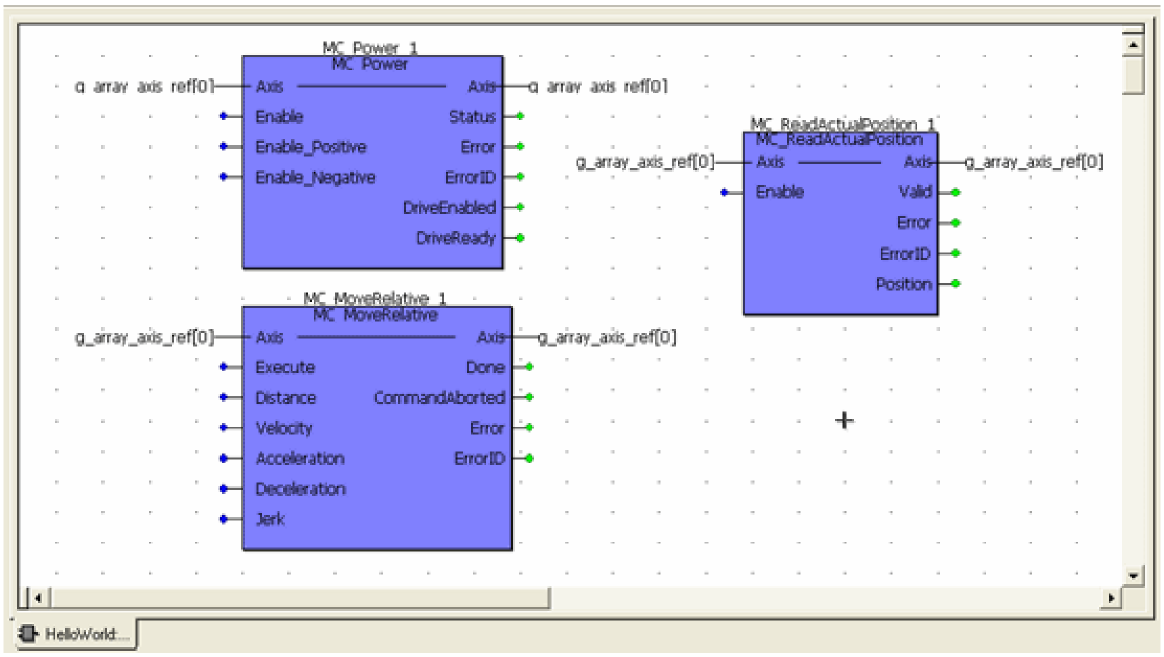


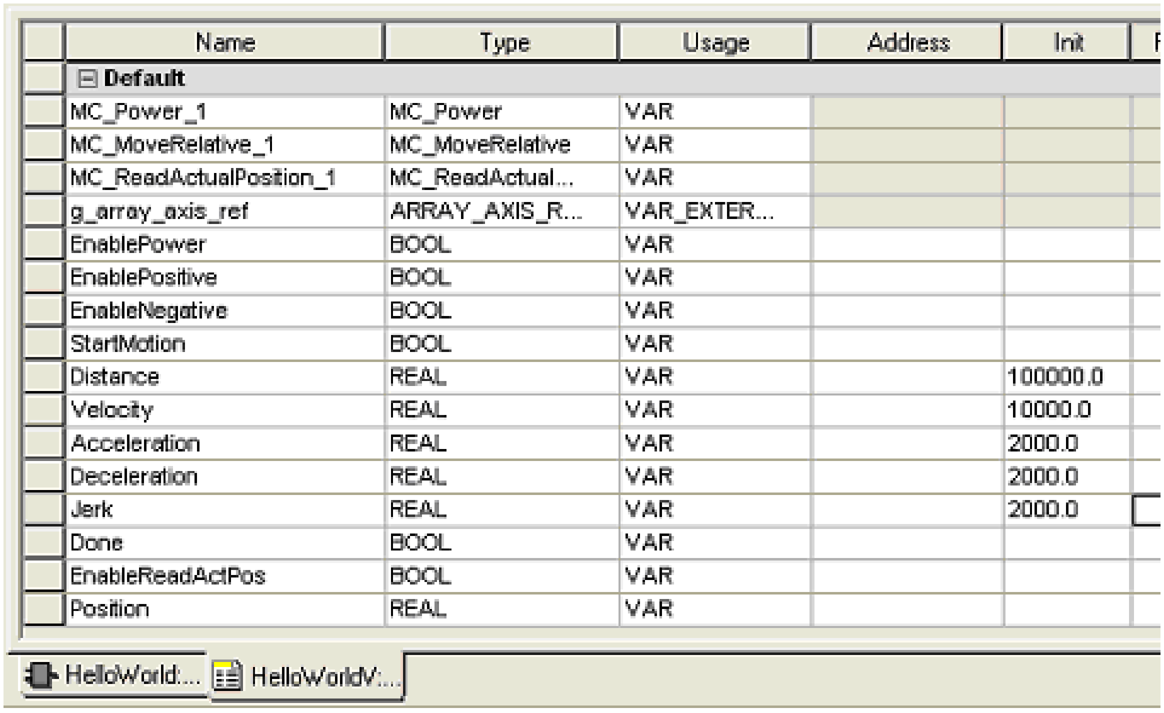
Fig. 21: Interfacing the first axis in the system to PLCopen FBs via g_array_axis_ref[0]

4.1.6 Step 6: Assigning the function block IN/OUT variables

Variables are connected to the required inputs/outputs of the PLCopen FB that can be written with values later, thus commanding motion. Refer to the table for the initialisation values:

Variables to link to inputs/outputs of PLCopen FBs

Variable	Data type	Initialisation value
MC_Power_1 instance		
EnablePower	BOOL	FALSE
EnablePositive	BOOL	FALSE
EnableNegative	BOOL	FALSE
MC_ReadActualPosition_1 instance		
EnableReadActPos	BOOL	TRUE
Position	REAL	
MC_MoveRelative_1 instance		
Distance	REAL	100000.0
Velocity	REAL	10000.0
Acceleration	REAL	2000.0
deceleration	REAL	2000.0
Jerk	REAL	2000.0
Done	BOOL	



Name	Type	Usage	Address	Init
⊟ **Default**				
MC_Power_1	MC_Power	VAR		
MC_MoveRelative_1	MC_MoveRelative	VAR		
MC_ReadActualPosition_1	MC_ReadActual...	VAR		
g_array_axis_ref	ARRAY_AXIS_R...	VAR_EXTER...		
EnablePower	BOOL	VAR		
EnablePositive	BOOL	VAR		
EnableNegative	BOOL	VAR		
StartMotion	BOOL	VAR		
Distance	REAL	VAR		100000.0
Velocity	REAL	VAR		10000.0
Acceleration	REAL	VAR		2000.0
Deceleration	REAL	VAR		2000.0
Jerk	REAL	VAR		2000.0
Done	BOOL	VAR		
EnableReadActPos	BOOL	VAR		
Position	REAL	VAR		

Fig. 22: Declaration of variables in the variable sheet

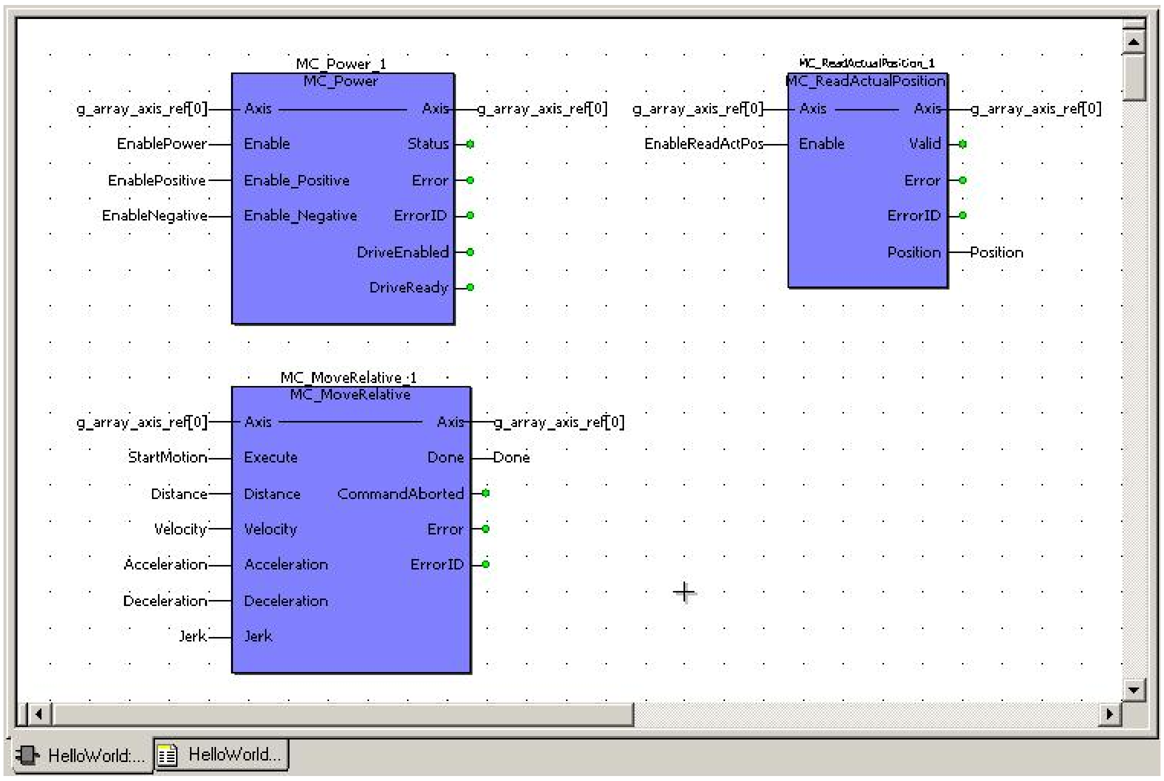
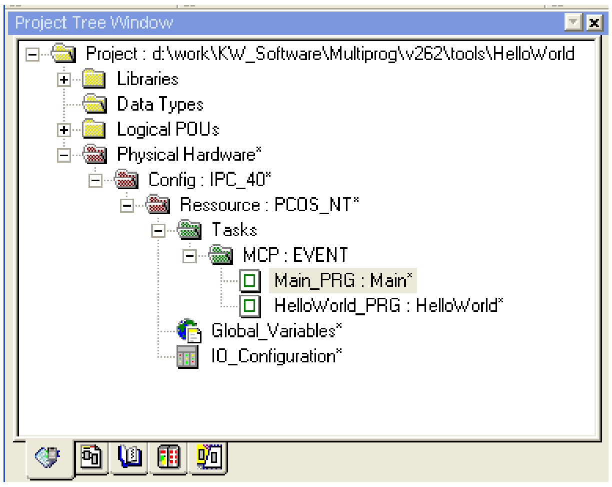


Fig. 23: Input/output variables linked to PLCopen-FBs

4.1.7 Step 7: Assigning programs to a task

Every program is assigned to an instance of a task. It is absolutely necessary for the Main program to be invoked before the HelloWorld application program since it invokes function blocks which are required for the MCE to function correctly.

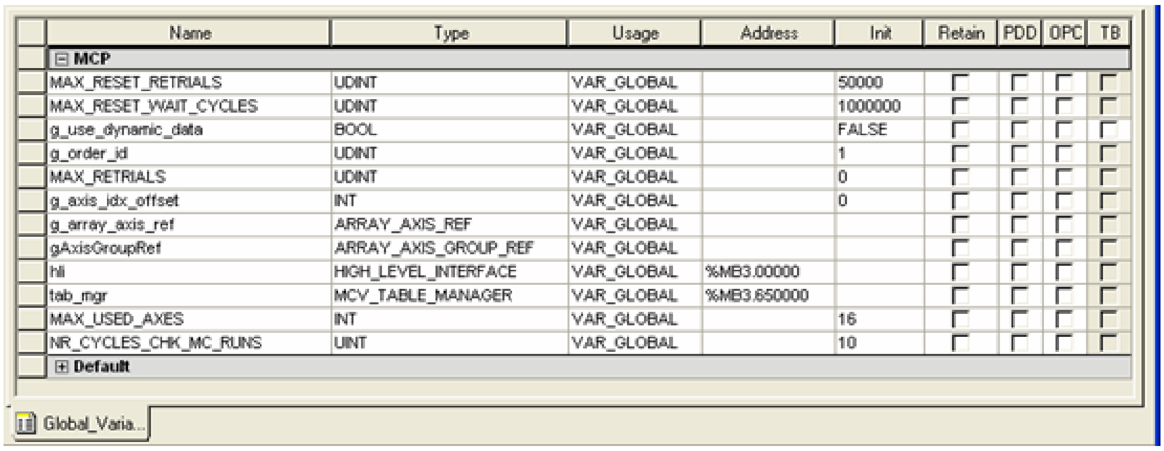


```
Project : d:\work\KW_Software\Multiprog\v262\tools\HelloWorld
  Libraries
  Data Types
  Logical POUs
  Physical Hardware*
    Config : IPC_40*
      Ressource : PCOS_NT*
        Tasks
          MCP : EVENT
            Main_PRG : Main*
            HelloWorld_PRG : HelloWorld*
        Global_Variables*
        IO_Configuration*
```

Fig. 24: Inserting instances of the Main and HelloWorld programs in a task

4.1.8 Step 8: Creating the required global variables

In the resource which also defines the task to invoke the two Main and HelloWorld programs, the required “Global Variables” must also be created, preferably in a separate group for greater clarity.



Name	Type	Usage	Address	Init	Retain	PDD	OPC	TB
MCP								
MAX_RESET_RETRIALS	UDINT	VAR_GLOBAL		50000	☐	☐	☐	☐
MAX_RESET_WAIT_CYCLES	UDINT	VAR_GLOBAL		1000000	☐	☐	☐	☐
g_use_dynamic_data	BOOL	VAR_GLOBAL		FALSE	☐	☐	☐	☐
g_order_id	UDINT	VAR_GLOBAL		1	☐	☐	☐	☐
MAX_RETRIALS	UDINT	VAR_GLOBAL		0	☐	☐	☐	☐
g_axis_idx_offset	INT	VAR_GLOBAL		0	☐	☐	☐	☐
g_array_axis_ref	ARRAY_AXIS_REF	VAR_GLOBAL			☐	☐	☐	☐
gAxisGroupRef	ARRAY_AXIS_GROUP_REF	VAR_GLOBAL			☐	☐	☐	☐
hli	HIGH_LEVEL_INTERFACE	VAR_GLOBAL	%MB3.00000		☐	☐	☐	☐
tab_mgr	MCV_TABLE_MANAGER	VAR_GLOBAL	%MB3.650000		☐	☐	☐	☐
MAX_USED_AXES	INT	VAR_GLOBAL		16	☐	☐	☐	☐
NR_CYCLES_CHK_MC_RUNS	UINT	VAR_GLOBAL		10	☐	☐	☐	☐
Default								

Fig. 25: Global variables are created in the corresponding resource

4.1.9 Step 9: Sending a project and cold start

After successful compiling of the project, it is sent to the resource and is started via the "Cold" button.

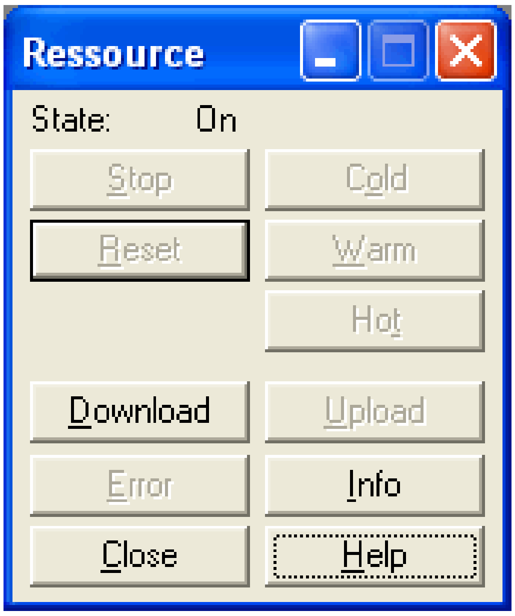


Fig. 26: Control dialog for sending, starting the PLC application

After the project starts, the individual values of the variables linked to the function blocks are displayed in the FDB.

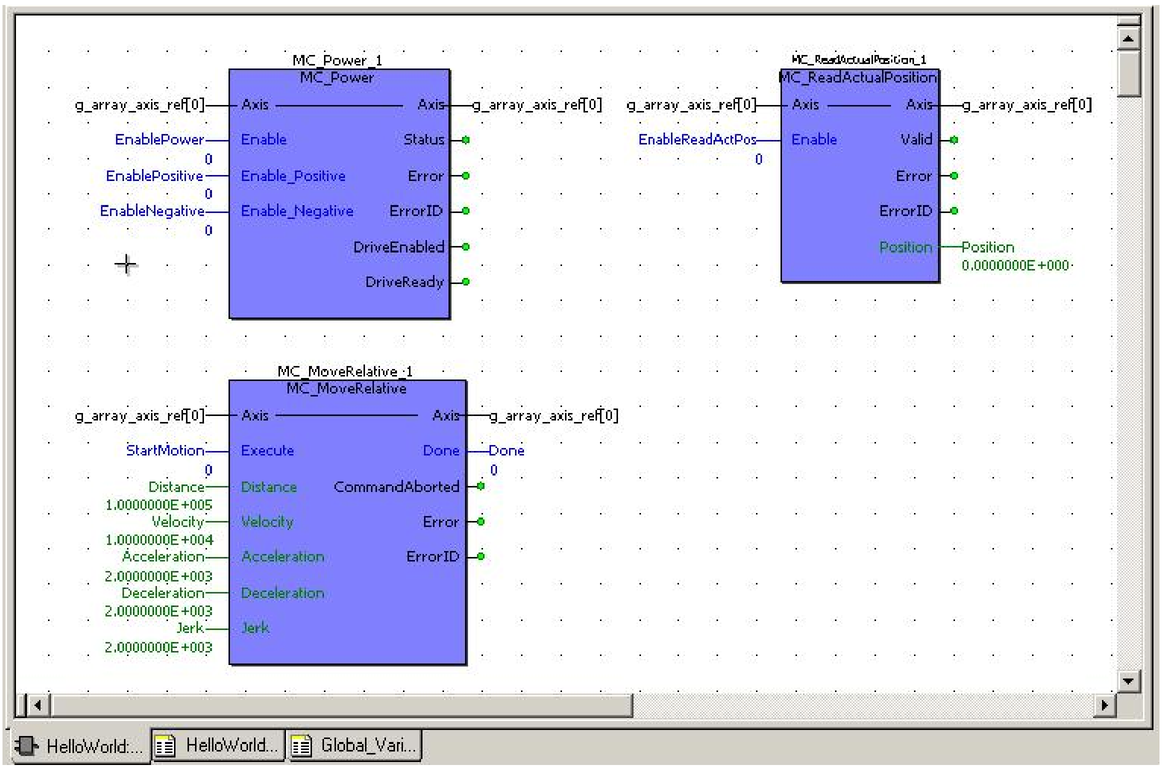


Fig. 27: State of the HelloWorld program after program start

4.1.10 Step 10: Setting the enabling signals for the axis

The values for the input/output variables can be overwritten in the debug mode. Controller and feed enabling is first issued for drive of the axis.

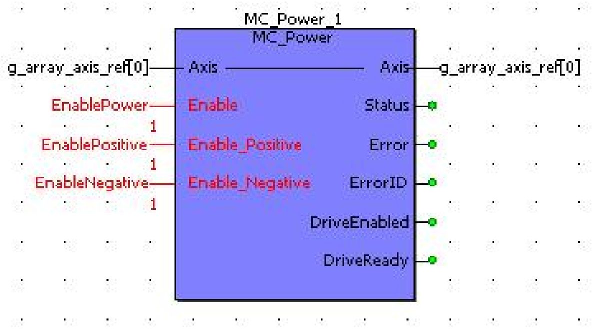
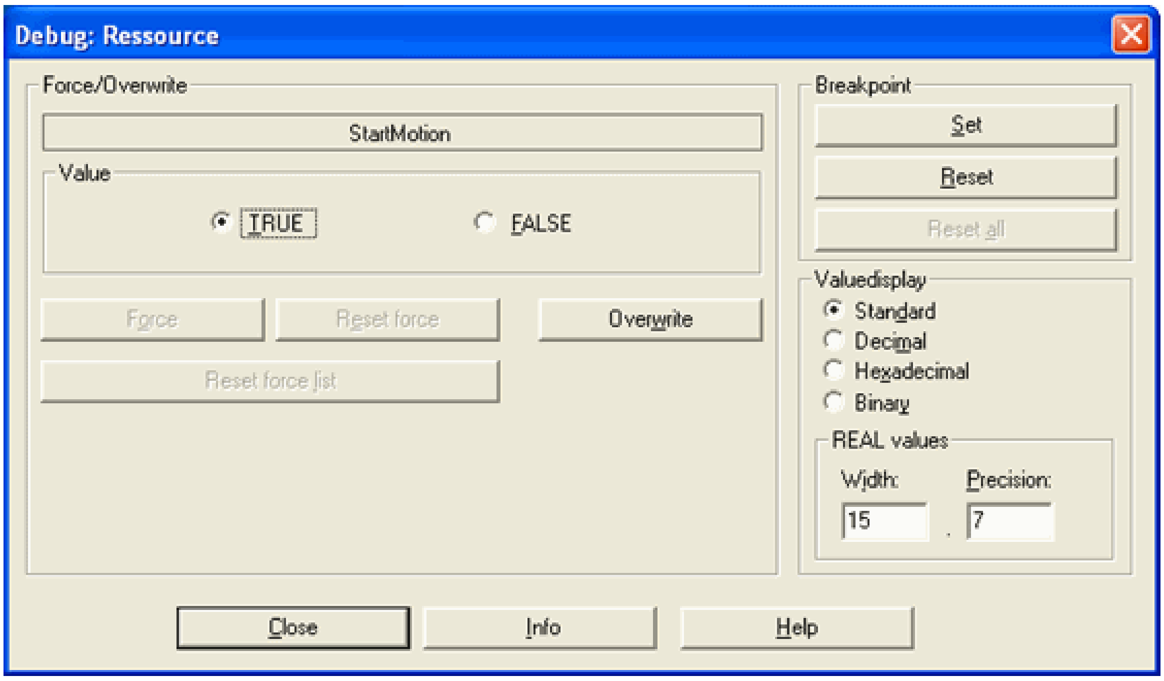


Fig. 28: Setting controller and feed release at MC_Power_1

4.1.11 Step 11: Setting the enabling signals for PLCopen FBs

To trigger motion, the **StartMotion** input variable must be set to **TRUE**.


Debug: Ressource

Force/Overwrite

StartMotion

Value: [✓ TRUE] FALSE

Valuedisplay: [✓ Standard] Decimal Hexadecimal Binary

REAL values

Width: 15

Precision: 7

Fig. 29: Setting the StartMotion input variable to start motion

4.1.12 Step 12: Ready, axis has moved!

After the StartMotion variable is set to TRUE, the motion starts with the first axis and the current actual position of the axis can be read at the FB MC_ReadActualPosition_1.

The end of the motion is shown when the output "Done“ is set at the FB MC_MoveRelative_1. This output remains TRUE until a falling edge is detected at the “Execute” input. This is obtained by resetting the variable to FALSE.

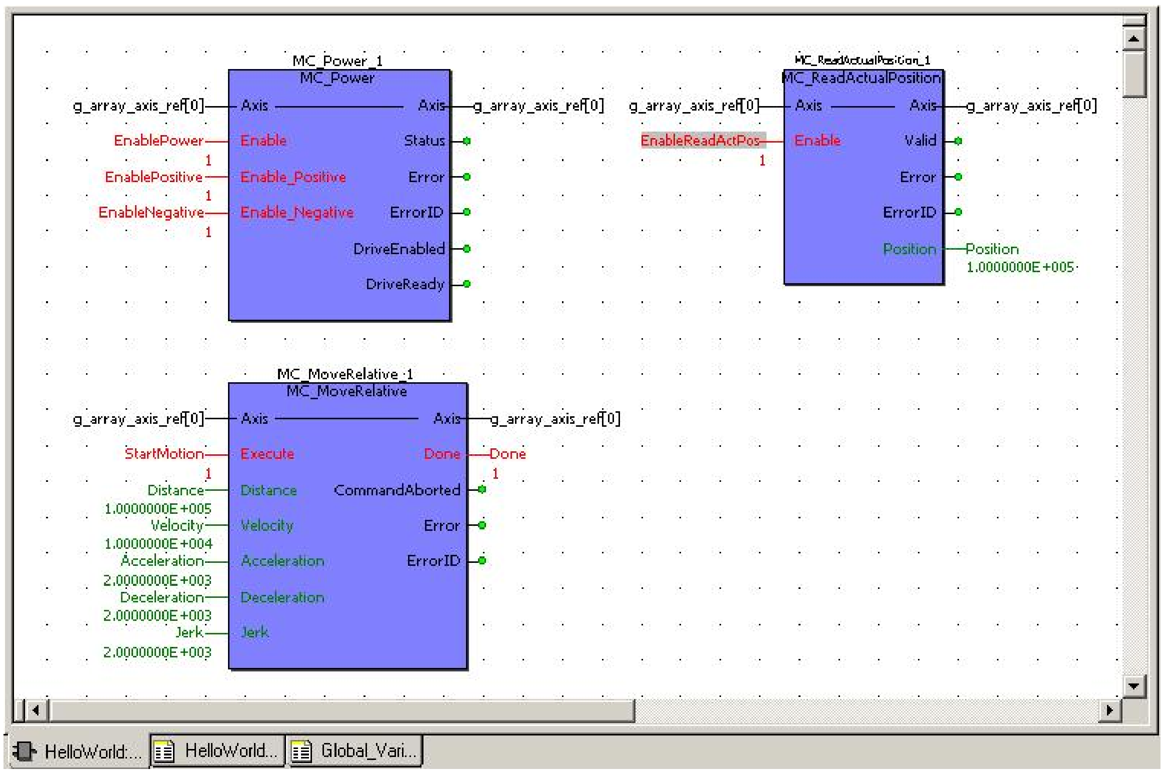


Fig. 30: Status after motion

4.2 "CoDeSys" programming example

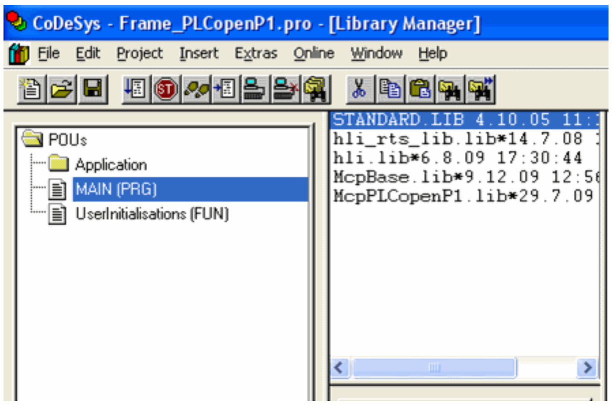
Based on the PLC project Frame_PLCopenP1.pro, the example shows how a simple motion task is executed. In this general application, all programs and function blocks are invoked in the MAIN program, set up links to the motion controller and initialise the PLC working data.

4.2.1 Step 1: Required libraries

To execute a motion task using PLCopen FB, the PLC libraries

- **STANDARD.LIB**, belongs to the PLC runtime system
- **hli_rts_lib.lib**, contains the utility FB
- **hli.lib**, simulates the memory between the PLC and the MC
- **McpBase.lib**, sets up a link to the MC; supplies data structures and FBs which are used in other libraries
- **McpPLCopenP1.lib**, FB according to PLCopen specifications Parts 1 and 2

integrated in the application. This is already the case in the example program **Frame_PLCopenP1.pro**.


CoDeSys - Frame_PLCopenP1.pro - [Library Manager]
File Edit Project Insert Extras Online Window Help
POUs
Application
MAIN (PRG)
UserInitialisations (FUN)

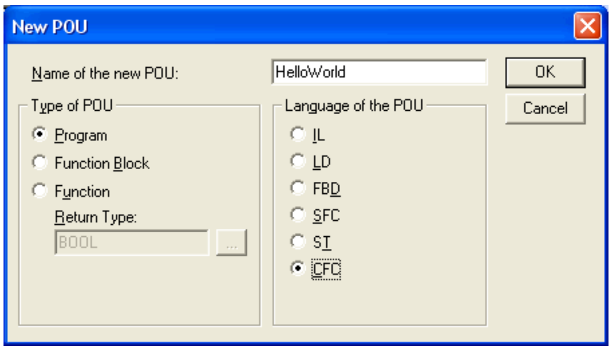
```
STANDARD.LIB 4.10.05 11:
hli_rts_lib.lib*14.7.08 1
hli.lib*6.8.09 17:30:44
McpBase.lib*9.12.09 12:5
McpPLCopenP1.lib*29.7.09
```

Fig. 31: Required libraries linked to the project

Open this example program and save it as HelloWorld.pro before executing the other steps to execute the task.

4.2.2 Step 2: Creating the HelloWorld application program

The motion task is implemented in the HelloWorld program. Use the free-graphic function block diagram editor (CFC) for implementation The program is created in the existing folder “Application“ in compliance with these specification:

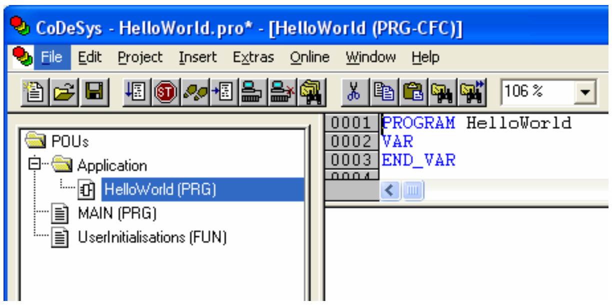

New POU

Name of the new POU: HelloWorld

Type of POU: [✓ Program] Function Block, Function; Return Type: BOOL

Language of the POU: IL, LD, FBD, SFC, ST, [✓ CFC]

Fig. 32: Definition of the HelloWorld program


CoDeSys - HelloWorld.pro* - [HelloWorld (PRG-CFC)]

POUs
- Application
  - HelloWorld (PRG)
- MAIN (PRG)
- UserInitialisations (FUN)

```
PROGRAM HelloWorld
VAR
END_VAR
```

Fig. 33: Assigning the Main and HelloWorld programs in the project tree

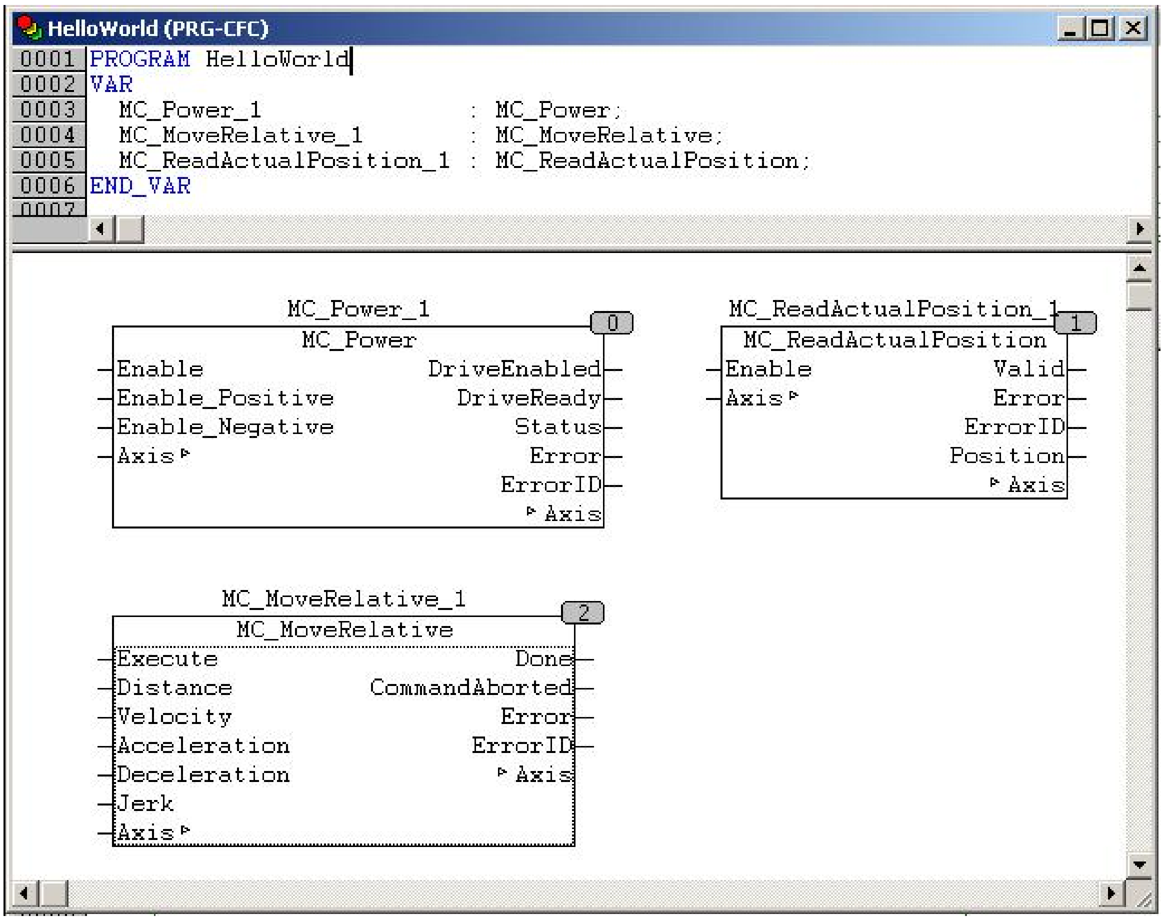
4.2.3 Step 3: HelloWorld program: Instancing PLCopen FBs

The FB instances listed below are required to execute the task and are created in the HelloWorld program in which the motion task is to be implemented:

Instances of the FBs used in the HelloWorld program

PLCopen FB	Instance name	Remarks
MC_Power	MC_Power_1	Serves to set controller and feed enabling
MC_ReadActualPosition	MC_ReadActualPosition_1	Shows the position of the axis at the OUT variable "position"
MC_MoveRelative	MC_MoveRelative_1	Moves the axis by the value of the IN variable "distance" relative to the current position.

The figure below shows the variables definition area in which the required function blocks are defined and how they are displayed in the CFC editor.

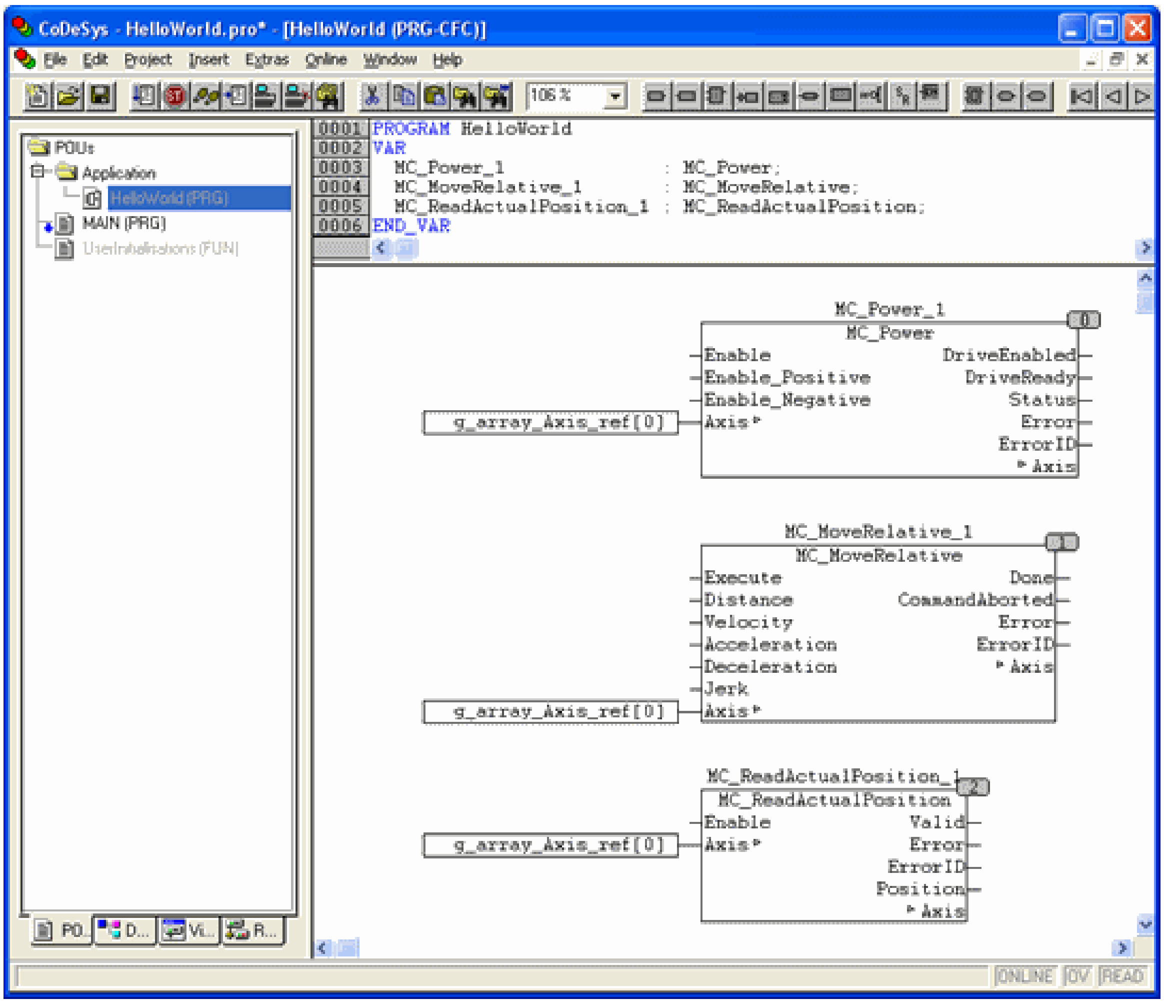


```
PROGRAM HelloWorld
VAR
  MC_Power_1                : MC_Power;
  MC_MoveRelative_1         : MC_MoveRelative;
  MC_ReadActualPosition_1   : MC_ReadActualPosition;
END_VAR
```

Fig. 34: Instances of PLCopen FBs in the HelloWorld world

4.2.4 Step 4: Linking the axes to PLCopen FBs

The axis reference g_array_axis_ref[0] which links to the first axis in the system is applied to all PLCopen FBs..



```
PROGRAM HelloWorld
VAR
    MC_Power_1                  : MC_Power;
    MC_MoveRelative_1           : MC_MoveRelative;
    MC_ReadActualPosition_1     : MC_ReadActualPosition;
END_VAR
```

Fig. 35: Interfacing the first axis in the system to PLCopen FBs via g_array_axis_ref[0]

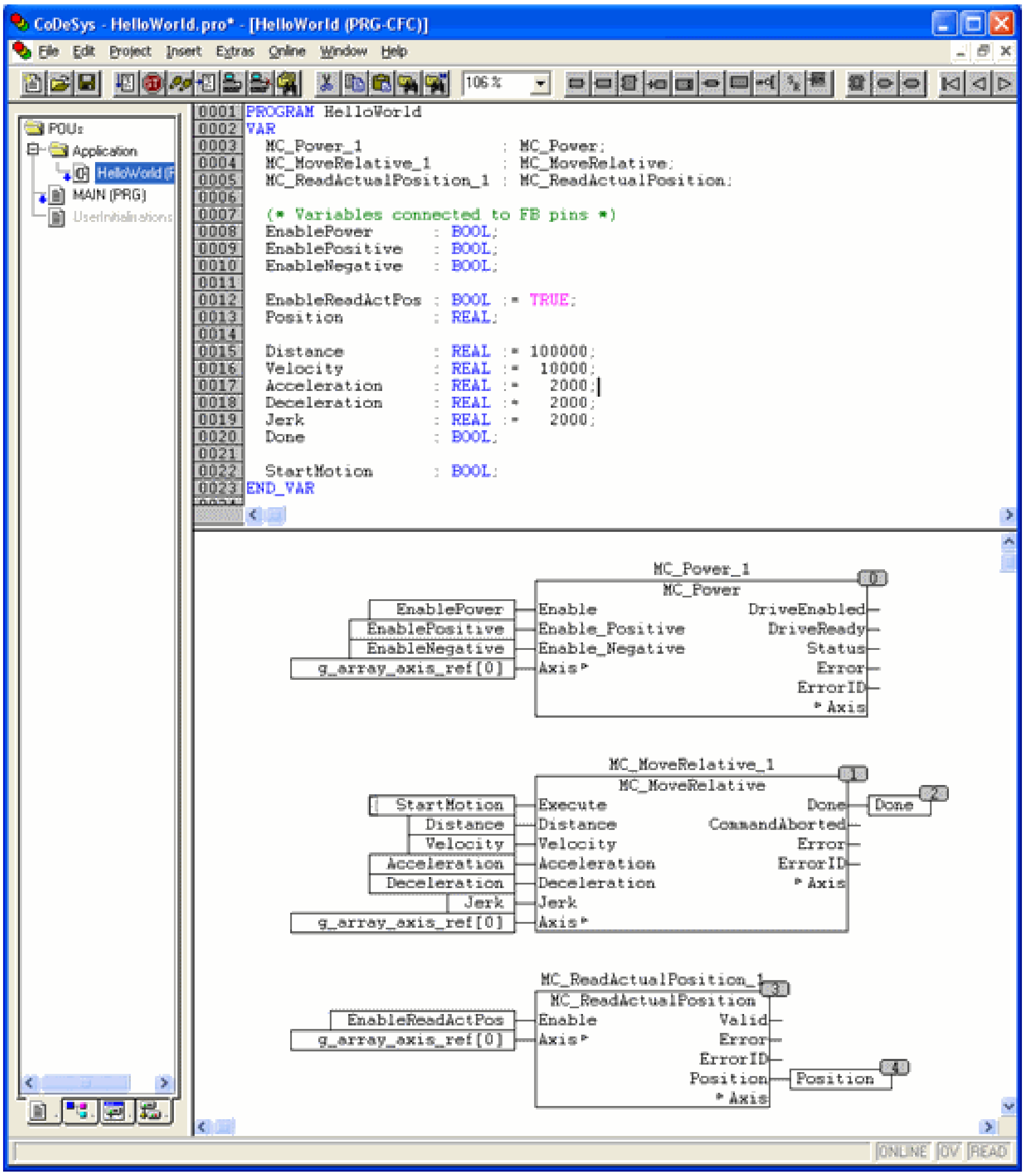
4.2.5 Step 5: Assigning the function block IN/OUT variables

Variables are connected to the required inputs/outputs of the PLCopen FB that can be written with values later, thus commanding motion. Refer to the table for the initialisation values:

Variables to link to inputs/outputs of PLCopen FBs

Variable	Data type	Initialisation value
MC_Power_1 instance		
EnablePower	BOOL	FALSE
EnablePositive	BOOL	FALSE
EnableNegative	BOOL	FALSE
MC_ReadActualPosition_1 instance		
EnableReadActPos	BOOL	TRUE
Position	REAL	
MC_MoveRelative_1 instance		
Distance	REAL	100000.0
Velocity	REAL	10000.0
Acceleration	REAL	2000.0
deceleration	REAL	2000.0
Jerk	REAL	2000.0
Done	BOOL	

The variables are created in the variables definition area and some of them are initialised with default values. The CFC editor shows that the variables are already linked to the corresponding input/output pins of the function blocks:

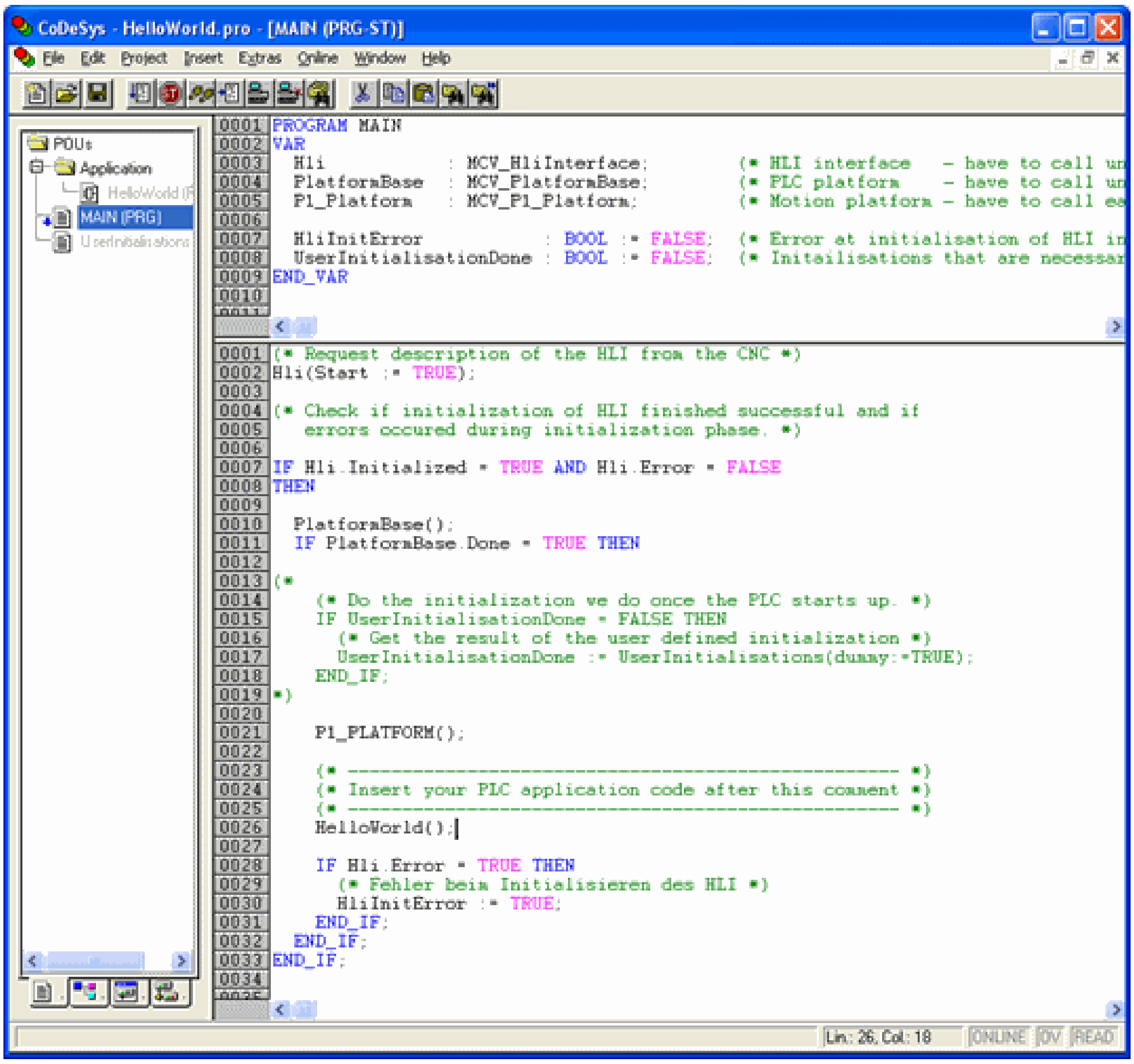


```
PROGRAM HelloWorld
VAR
  MC_Power_1              : MC_Power;
  MC_MoveRelative_1       : MC_MoveRelative;
  MC_ReadActualPosition_1 : MC_ReadActualPosition;

  (* Variables connected to FB pins *)
  EnablePower          : BOOL;
  EnablePositive       : BOOL;
  EnableNegative       : BOOL;

  EnableReadActPos     : BOOL := TRUE;
  Position             : REAL;

  Distance             : REAL := 100000;
  Velocity             : REAL :=  10000;
  Acceleration         : REAL :=   2000;
  Deceleration         : REAL :=   2000;
  Jerk                 : REAL :=   2000;
  Done                 : BOOL;

  StartMotion          : BOOL;
END_VAR
```

Fig. 36: Input/output variables linked to PLCopen-FBs

4.2.6 Step 6: Insert the HelloWorld program in the MAIN program

The MAIN program was already created in the Frame_PLCopenP1 example application.pro and contains all the function blocks and their invocation required for the MCE to function correctly. To execute the HelloWorld program, it is added after the comment "Insert your PLC application code after this comment".


CoDeSys - HelloWorld.pro - [MAIN (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
Application
HelloWorld (P
MAIN (PRG)
UserInitialisations
Lin.: 26, Col.: 18
ONLINE
OV
READ

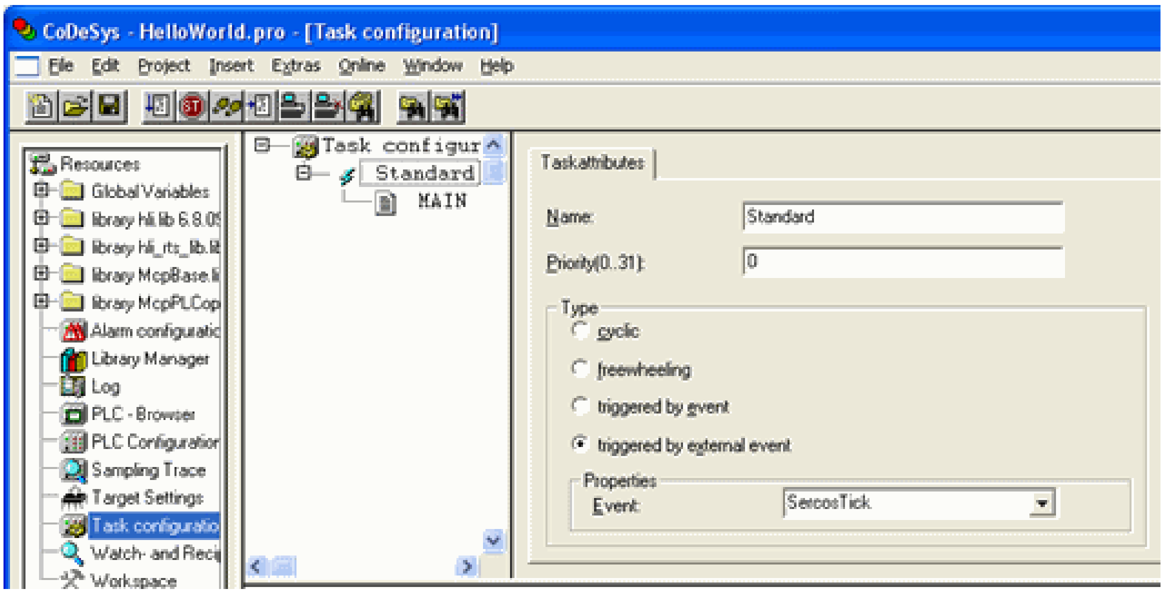
```
PROGRAM MAIN
VAR
  Hli                    : MCV_HliInterface;        (* HLI interface    - have to call un
  PlatformBase           : MCV_PlatformBase;        (* PLC platform     - have to call un
  P1_Platform            : MCV_P1_Platform;         (* Motion platform  - have to call ea

  HliInitError                : BOOL := FALSE;   (* Error at initialisation of HLI in
  UserInitialisationDone      : BOOL := FALSE;   (* Initailisations that are necessar
END_VAR
```

```
(* Request description of the HLI from the CNC *)
Hli(Start := TRUE);

(* Check if initialization of HLI finished successful and if
   errors occured during initialization phase. *)
IF Hli.Initialized = TRUE AND Hli.Error = FALSE
THEN

  PlatformBase();
  IF PlatformBase.Done = TRUE THEN

(*
    (* Do the initialization we do once the PLC starts up. *)
    IF UserInitialisationDone = FALSE THEN
      (* Get the result of the user defined initialization *)
      UserInitialisationDone := UserInitialisations(dummy:=TRUE);
    END_IF;
*)

    P1_PLATFORM();

    (* ---------------------------------------------------- *)
    (* Insert your PLC application code after this comment *)
    (* ---------------------------------------------------- *)
    HelloWorld();

    IF Hli.Error = TRUE THEN
      (* Fehler beim Initialisieren des HLI *)
      HliInitError := TRUE;
    END_IF;
  END_IF;
END_IF;
```

Fig. 37: Insert the HelloWorld program in the MAIN program

4.2.7 Step 7: Assigning programs to a task

The MAIN program is already assigned to a task in the example application Frame_PLCopenP1.pro. The figure below shows the corresponding settings:



Name: Standard

Priority(0..31): 0

Type
- cyclic
- freewheeling
- triggered by event
- [✓ triggered by external event]

Properties
Event: SercosTick

Fig. 38: The MAIN program is assigned to the Standard task

4.2.8 Step 8: Build application, login, run

Building and logging the application then take place. The initial values can be seen in the variables after the application starts.

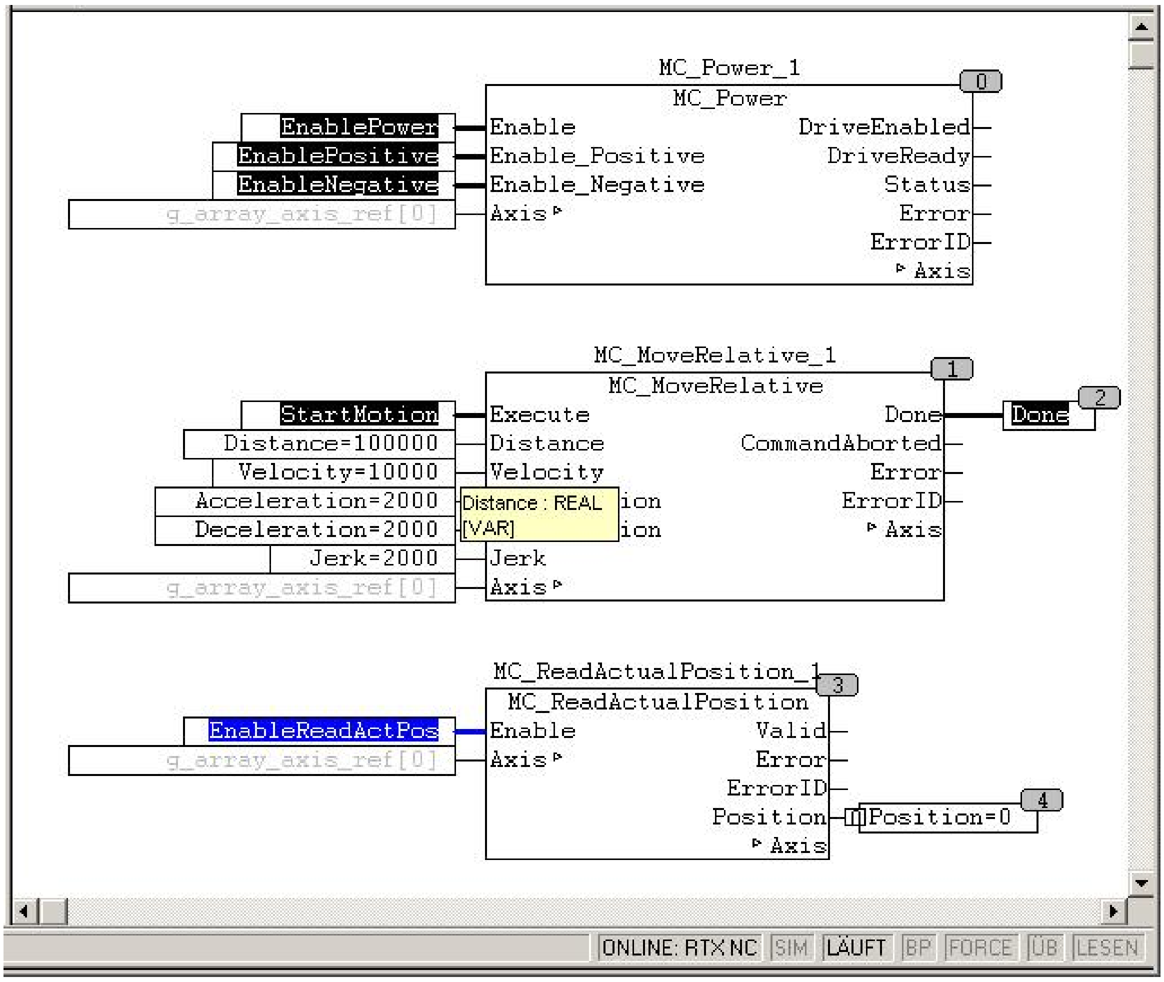


Fig. 39: HelloWorld after the application starts

4.2.9 Step 9: Setting the enabling signals for the axis

Controller and feed enabling is first issued for drive of the axis. This takes place by overwriting or forcing input variables at the function block MC_Power_1.

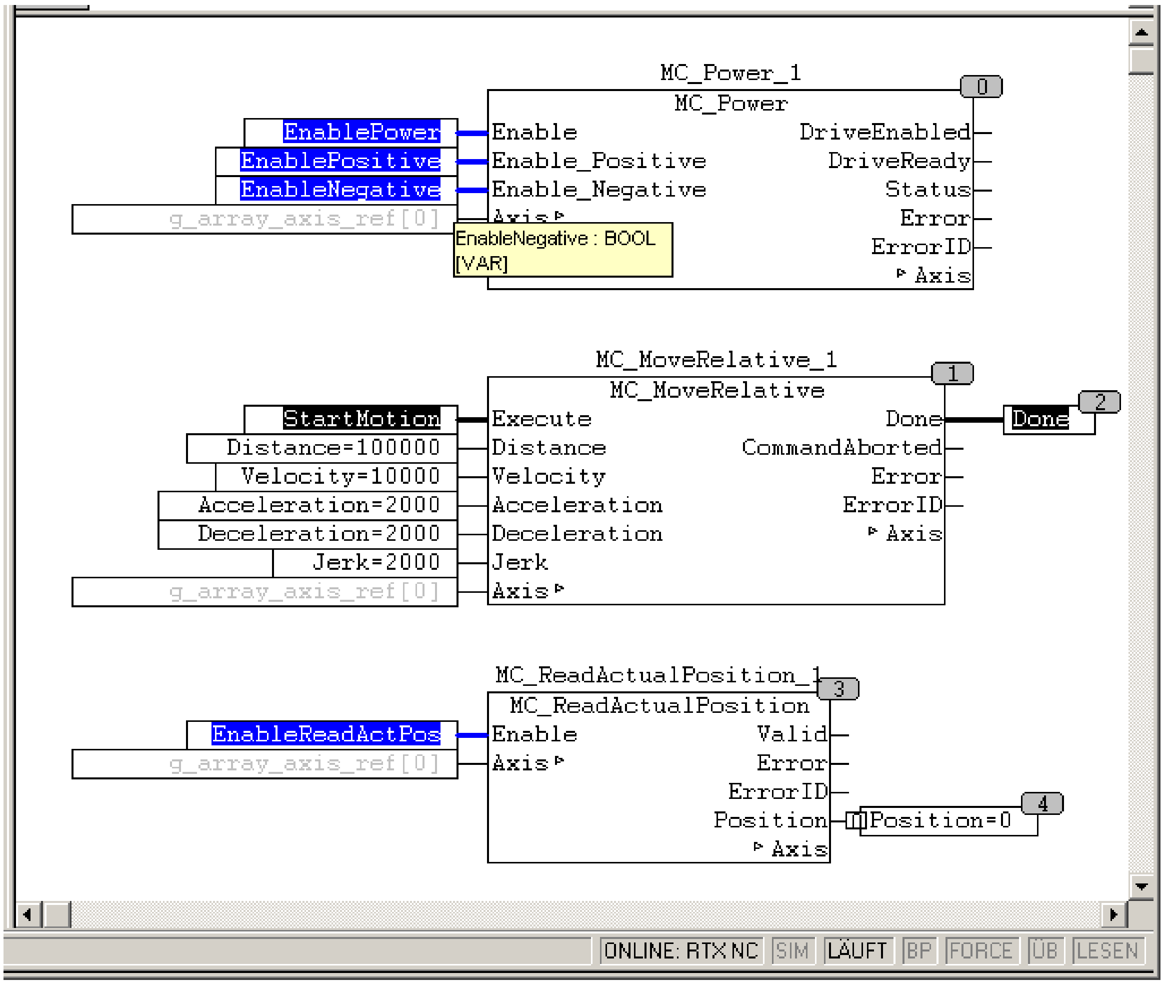


Fig. 40: Setting controller and feed release at MC_Power_1

4.2.10 Step 10: Ready, axis has moved!

The axis motion is triggered by setting the value of the StartMotion variable in the function block MC_Move_Relative_1 to value TRUE. The current actual position of the axis can then be read in the “Position” variable at the FB MC_ReadActualPosition_1.

The figure below shows the state of the function blocks and variables at the end of the motion. The end of the motion is clearly visible since the “Done” variable now has the value TRUE. This output remains TRUE until a falling edge is detected at the “Execute” input. This is obtained by resetting the variable to FALSE.

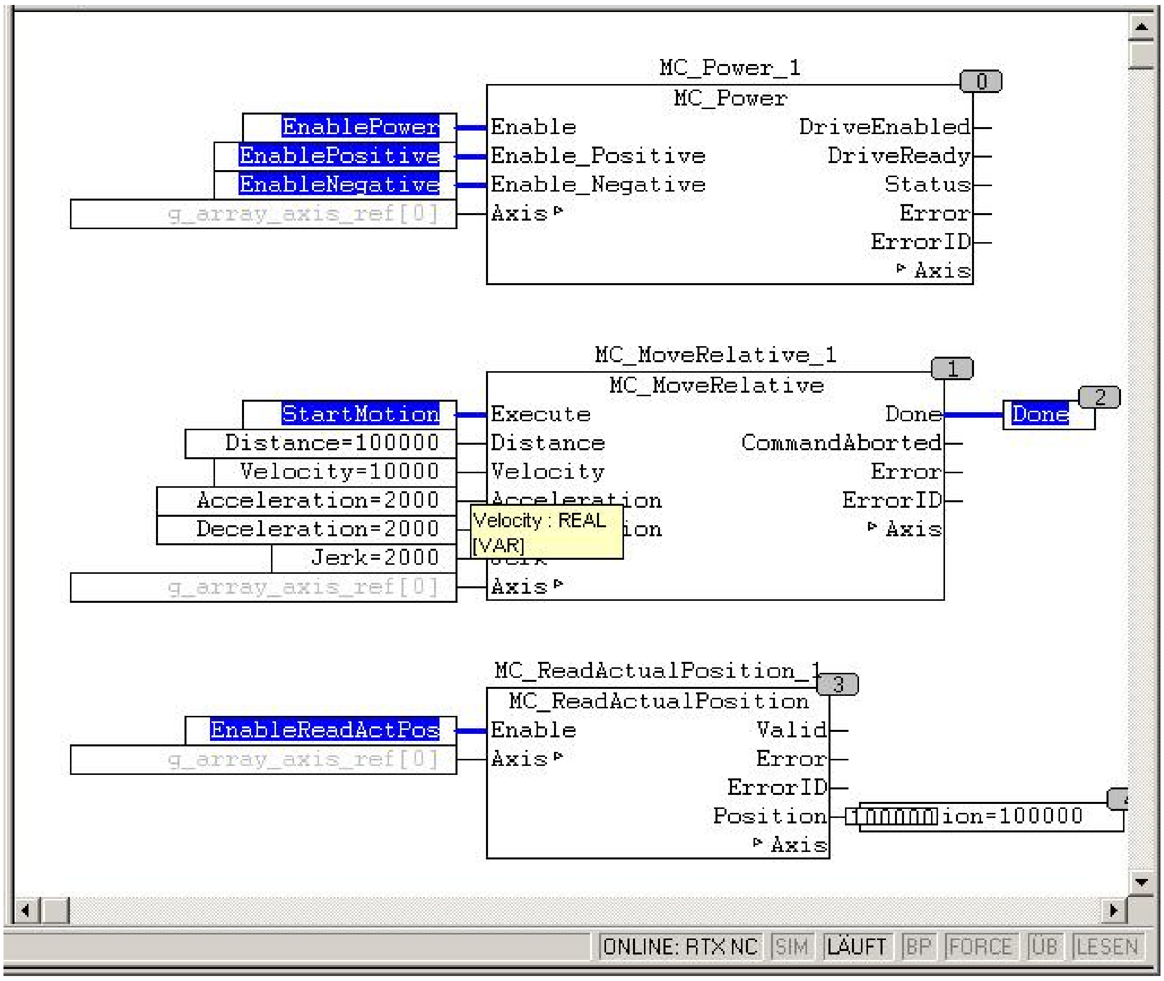


Fig. 41: State at end of motion

5 References

[1] PLCopen specifications: TC2 Task Force Motion Control “Function Blocks for motion control” Version 1.0, dated 23 Nov. 2001

[2] CNC PLC overall control system documentation

[3] The PLCopen Compliance Statement V1.0 from ISG can be found on the PLCopen website (www.plcopen.org).

6 Appendix

6.1 Suggestions, corrections and the latest documentation

Did you find any errors? Do you have any suggestions or constructive criticism? Then please contact us at documentation@isg-stuttgart.de. The latest documentation is posted in our Online Help (DE/EN):

QR code link: https://www.isg-stuttgart.de/documentation-kernel/

The link above forwards you to:

https://www.isg-stuttgart.de/fileadmin/kernel/kernel-html/index.html

Notice

Change options for favourite links in your browser;

Technical changes to the website layout concerning folder paths or a change in the HTML framework and therefore the link structure cannot be excluded.

We recommend you to save the above "QR code link" as your primary favourite link.

PDFs for download:

DE:
https://www.isg-stuttgart.de/produkte/softwareprodukte/isg-kernel/dokumente-und-downloads

EN:
https://www.isg-stuttgart.de/en/products/softwareproducts/isg-kernel/documents-and-downloads

E-Mail: documentation@isg-stuttgart.de

Index