DOCUMENTATION ISG-kernel

PLC library
PLC Examples

Short Description:
MCP-APEX

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de

Documentation version: 1.03
13/12/2023

Preface

Legal information

This documentation was produced with utmost care. The products and scope of functions described are under continuous development. We reserve the right to revise and amend the documentation at any time and without prior notice.

No claims may be made for products which have already been delivered if such claims are based on the specifications, figures and descriptions contained in this documentation.

Personnel qualifications

This description is solely intended for skilled technicians who were trained in control, automation and drive systems and who are familiar with the applicable standards, the relevant documentation and the machining application.

It is absolutely vital to refer to this documentation, the instructions below and the explanations to carry out installation and commissioning work. Skilled technicians are under the obligation to use the documentation duly published for every installation and commissioning operation.

Skilled technicians must ensure that the application or use of the products described fulfil all safety requirements including all applicable laws, regulations, provisions and standards.

Further information

Links below (DE)

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

or (EN)

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

contains further information on messages generated in the NC kernel, online help, PLC libraries, tools, etc. in addition to the current documentation.

Disclaimer

It is forbidden to make any changes to the software configuration which are not contained in the options described in this documentation.

Trade marks and patents

The name ISG®, ISG kernel®, ISG virtuos®, ISG dirigent® and the associated logos are registered and licensed trade marks of ISG Industrielle Steuerungstechnik GmbH.

The use of other trade marks or logos contained in this documentation by third parties may result in a violation of the rights of the respective trade mark owners.

Copyright

General and safety instructions

Icons used and their meanings

This documentation uses the following icons next to the safety instruction and the associated text. Please read the (safety) instructions carefully and comply with them at all times.

Icons in explanatory text

➢ Indicates an action.

⇨ Indicates an action statement.

⚠ DANGER

Acute danger to life!

If you fail to comply with the safety instruction next to this icon, there is immediate danger to human life and health.

⚠ CAUTION

Personal injury and damage to machines!

If you fail to comply with the safety instruction next to this icon, it may result in personal injury or damage to machines.

Attention

Restriction or error

This icon describes restrictions or warns of errors.

Notice

Tips and other notes

This icon indicates information to assist in general understanding or to provide additional information.

Example

General example

Example that clarifies the text.

Programing Example

NC programming example

Programming example (complete NC program or program sequence) of the described function or NC command.

Release Note

Specific version information

Optional or restricted function. The availability of this function depends on the configuration and the scope of the version.

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 HLI libraries

ISG supplies a number of PLC libraries which are linked by the applications described. Since ISG supports a number of different PLC runtime systems running on various operating systems and in different control units, this chapter lists the individual libraries and agrees on a standard name. This is used in this documentation to apply a standard name for libraries of different names which have the same content.

Since **TwinCAT 3** was introduced, there are other variants of the libraries. The libraries are distinguished by the prefix in front of the library name:

- CNC Build = 28xx: Prefix = **pf_** = TcCnc
- CNC Build > 3xxx: Prefix = **pf_** = Tc2_Cnc

Overview of PLC libraries

Standard name	**3S / CoDeSys**	**TwinCAT**	**KW/Multiprog**
HLI library	hli.lib or 00_CNCHLIV**X_XXXX**.lib	**CNC Build = 28xx** **pf_**HliV3.lib **CNC Build = 3xxx** **pf_**Hli.lib	hli.mwt
Base motion library	McpBase.lib	**pf_**Base.lib	McpBase.mwt
Part 1 motion library	McpPlcopenP1.lib	**pf_**PlcopenP1.lib	McpPlcopenP1.mwt
Part 4 motion library	McpPlcopenP4.lib	**pf_**PlcopenP4.lib	McpPlcopenP4.mwt
Techno library	McpTechTct.lib	**pf_**TechTct.lib	McpTechTct.mwt
SERCOS library	McpSercos.lib		
Adaption library	**See the table below**		

Variants of adaption library

3S / CoDeSys		
RTX	**Win32**	**Linux/Homag**
hli_rts_lib.lib	hli_rts_lib.lib	00_CNCHLIV**X**INIT_**XXXX**.lib

3 Frame_PLCopenP1.pro: General application for the use of PLCopen function blocks

This example program shows which function blocks must be created so that an application functions properly and the function blocks which are to be used in compliance with PLCopen Part1 and Part4.

3.1 Required libraries

The following libraries must be integrated in this application:

- Adaption library for RTX or Win32 simulation mode
- HLI library
- Base motion library
- Part 1 motion library

3.2 MAIN() main program

In the MAIN program an instance of the FB MCV_HliInterface from the HLI library is first calculated. This FB checks whether the HLI interface on the PLC side corresponds to the interface on the motion controller side. Here the FB must be calculated several times. Only when the output **Initialized** of the FB is TRUE and the output **Error** indicates FALSE can access be made to the HLI, the interface between the PLC and the motion controller.

For this reason an instance of the FB MCV_PlatformBase may only be invoked later. Axis references are initialised with this FB. Only after the FB signals by the **Done** output = TRUE that the axis references can be used, can an instance of the FB MCV_P1_PLATFORM be calculated. This FB must be invoked cyclically in the PLC application. The FB ensures that error messages are received by the motion controller and are forwarded to every FB via the axis references. This permits every FB to respond to problems with the axis which it commands.

The application code is inserted after this FB.

The MAIN() program is then assigned to a task.

4 PLCopen.pro: Test application for PLCopen function blocks

This application helps the user to familiarise himself with the response of function blocks according to the PLCopen standards Part1 and Part4. The application is designed to calculate all instances of these function blocks cyclically. There are no dependencies between function blocks. The user is provided with a command interface in the form of created visualisations. One visualisation was created for each function block type.

The concept in this example shows that each axis or axis group is always assigned the same function block instance. This helps to explain the example more easily. The same number of function blocks is created as the number of axes or axis groups to be commanded.

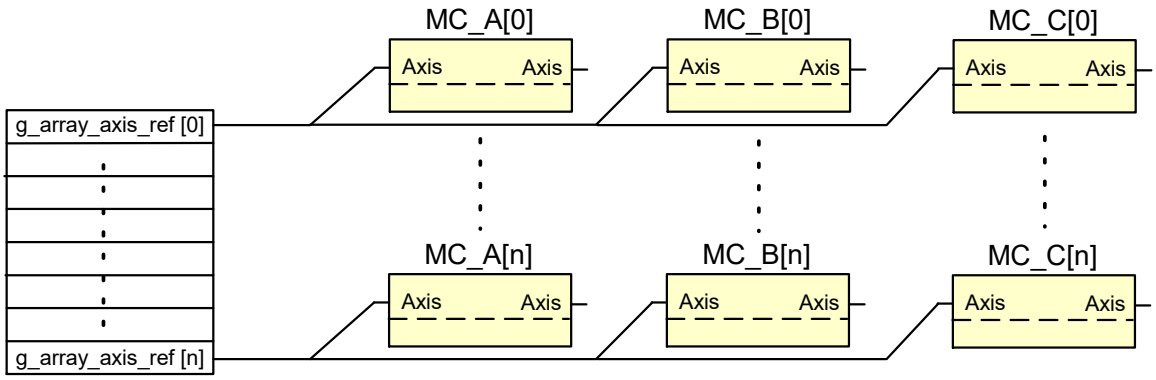


Fig. 1: Assigning axis or axis group references to function blocks

In order to provide proof that the functionality of function blocks is executed properly even when axes and axis groups are replaced, replacement can be authorised by 2 global variables (gP1AxisRefChgDisable, gP4AxesGrpRefChgDisable). However, this operating mode is conceived for demonstration purposes.

The example contains a single task which invokes the MAIN() main program.

4.1 Configuration requirements

This application assumes that the configured axes are created as axes with a separate interpolator (SAI). In this case it is sufficient to create the axes as spindles in the axis parameter

- lists: kenngr.achs_typ 4

or

- to activate the individual axis interpolator for other axis types by kenngr.configure_sai 1

All axes which are to be commanded by function blocks according to PLCopen specification Part 4 must be made known at the time of configuring the axis groups and must therefore be entered in the channel parameter lists.

4.2 Required libraries

The following libraries must be integrated in this application:

- Adaption library for RTX or Win32 simulation mode
- HLI library
- Base motion library
- Part 1 motion library
- Part 4 motion library
- Techno library
- SERCOS library

4.3 MAIN() main program

The MAIN program is based on the example of Frame_PlcopenP1.pro.

Since function blocks according to PLCopen specification Part 4 are also instances in this application, the platform module must be instanced and calculated for Part 4 function blocks. It is an MCV_P4_Platform type, contains the identifier P4_Platform and is invoked for Part 1 function blocks immediately after the platform function block.

The FB instances TechFctChIn and TechFctAxIn are used to notify technology information to the PLC via M, H, S or T commands from the NC program, to take this information from the HLI interface between the motion kernel and the PLC and to supply it in the PLC. The corresponding function blocks TechFctChOut and TechFctAxOut at the end of the MAIN program automatically acknowledge all technology functions which were not handled by the PLC project. To handle a technology function, an FB specified for the technology function type is instanced in the PLC project and supplied with the identification number of the technology function. This option is not highlighted in this example project. Instead only all incoming technology functions should be acknowledged so that an NC program can be executed without interruption.

After reading the technology functions, the programs are invoked successively. Ultimately they contain function blocks which enable the execution of motion tasks of individual axes or axis groups.

4.4 General implementation notes

Most programs described below are based on the same principle. In the initialisation phase, every function block instance is assigned to an axis or axis group required for operation by a variable of the AXIS_REF or AXES_GROUP_REF type. These variables are already created as an array in the motion base library (g_array_axis_ref and gAxesGroupRef). After this phase, all function block instances are calculated in a FOR loop in every PLC cycle.

Since only one visualisation is created and instanced for each function block type, it is also possible to switch this over to the various function block instances via the visualisation (see Visualisations for function blocks). The remaining code is required in the program in order to execute this process which contains pointers to function block instances.

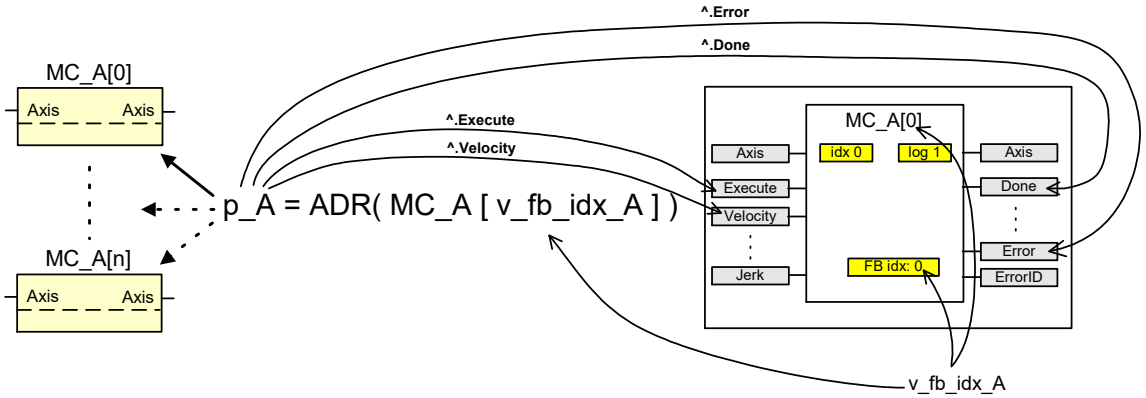


Fig. 2: Relationship between function blocks and visualisation

4.4.1 Global variables

Function blocks which are invoked by various programs are created as global variables and organised in function block arrays. The individual function block instances are also initialised at this point.

The variables required to supply the visualisations are also created as global variables.

4.4.2 Defaults values in PLCopen

Basically, default values are created as global constants for acceleration, deceleration, velocity and jerk and are used to supply the input variables of function blocks with suitable default values. Other explanations are contained in the table below. The remaining constants scale the application for the number of function blocks which are to be created and calculated for each function block type.

Default values in PLCopen.pro

Value	Description
2000.0	Default value for acceleration
2000.0	Default value for deceleration
10000.0	Default value for jerk
25000.0 50000.0 75000.0 100000.0	Default values for velocity
15	Number of function block instances created to command individual axes for function block types. Multiple axes can then be commanded at the same time when each axis is assigned exactly one function block instance.
8	Number of function block instances created to command master/slave axis pairs for function block types. Multiple slave axes can then be commanded at the same time when each axis is assigned exactly one function block instance.
9	Number of function block instances created to command axis groups for function block types. Multiple axis groups can then be commanded at the same time when each axis group is assigned exactly one function block instance.

4.5 Part 1 PLCopen program

This program is used to calculate function block instances that were implemented in accordance with Part 1 of the PLCopen specifications

In the initialisation part, the relationship between axis references and function blocks shown in the figure in "PLCopen.pro: Test application for PLCopen function blocks [▶ 11]" is re-established for function blocks referring to a single axis.

```
FOR Idx := 0 TO PLC_AX_MAXIDX DO
  IdxSingleAxFbs := Idx + g_axis_idx_offset;

  IF (IdxSingleAxFbs >= 0) AND
       (IdxSingleAxFbs <= PLC_AX_MAXIDX) THEN
    HomeAxRefIdx[IdxSingleAxFbs]     := IdxSingleAxFbs;
    …
       (* Calculate the administrative FBs *)
    ReadStatusAxRefIdx[IdxSingleAxFbs] := IdxSingleAxFbs;
      …
  END_IF;
END_FOR;
```

The first SAI axis available in the system is defined as the master axis for all function blocks that refer to a master and a slave axis. However, the individual function block instances then receive different references for the slave axis.

```
FOR IdxMultAxFbs := 0 TO PLC_MULTIAX_IDX DO
  …
  GearInMstIdx[IdxMultAxFbs]  := IdxMasterAx;
  GearInSlvIdx[IdxMultAxFbs]  := IdxSlaveAx;
  GearOutSlvIdx[IdxMultAxFbs] := IdxSlaveAx;
  …
  IF IdxSlaveAx < GC_MCP_AXREF_MAXIDX THEN
    IdxSlaveAx := IdxSlaveAx + 1;
  ELSE
    IdxMasterAx := IdxMasterAx + 1;
     IdxSlaveAx := 0;
  END_IF;
END_FOR;
```

After the initialisation phase, only the function block instances are calculated cyclically.

All the program code after the comment

```
(*=========================================================*)
(* The following commands are only for visualisation.      *)
(*=========================================================*)
```

is only used to supply the assigned visualisation with the data of the selected function block instance.

4.6 Part 1 MCV_Fbs program

This program requires no initialisations. Therefore, only the FBs defined by ISG and additionally supplied in the motion library Part 1 are calculated in the FOR loop. The code to supply and link the visualisation is then executed.

4.7 Part1_TouchProbe program

This program is used to calculate the instances of function blocks of type MC_AbortTrigger and MC_TouchProbe which supply the measuring function. These function blocks have a reference to the signal source of the TRIGGER_REF type as VAR_IN_OUT pin. This data type is defined in the Part 1 motion library. This type of reference is created for every FB instance in the initialisation part of the program and assigned suitable default values.

All FB instances of the types listed above are calculated in the FOR loop after the initialisation phase.

The remaining code is only used to supply the visualisation with data of the selected FB instance, as already executed for other programs.

4.8 Part4_PLCopen program

This program contains variables representing the ..._REF data types (IDENT_IN_GROUP_REF, MC_PATH_DATA_REF, MC_PATH_REF) from the specification. They are created as local variables and organised in arrays.

The 1:1 relationship of the indices of the function block instance and the axis groups is created in the initialisation part. In addition, several input variables of function blocks are initialised so that the user can obtain practical values.

This is followed by the well-known FOR loop where all function block instances are calculated. Then comes the code to update the data required for display in visualisations.

4.9 MotionOfAxes program

This program defines local variables in which the current position and selected state data of individual axes are saved after they are fetched from the HLI interface between the PLC and the motion kernel. The MotionOfAxes and MotionOfAxes_XY visualisations then indicate the values of these local variables.

4.10 MCV_Table_access program

This program is defined in the Part 1 motion library and permits access to cam tables. Data supplied by the available cam tables is displayed in the visualisation Tbm_Display. This data is supplied by the MCV_Table_access program.

4.11 MCV_TriggerCamIn program

4.12 Trace program

Calculates a function block instance which can initiate the generation of various logs in the motion kernel.

4.13 SERCOS_PRG program

This program commands one instance for each of the function blocks which write and read SERCOS parameters. The function block instances are then created locally in the program.

4.14 Visualisations for function blocks

A visualisation is created in the application for each of the implemented function block types. This visualisation can change the input variable values for every instance of a function block type and display the values of the output variables. In addition to the elements which represent the input and output variables according to the specifications, there are also several elements which are important to operate the visualisation. Their function is briefly described below:

Elements in visualisations for function blocks

No.	Description	Description
1	Function block index	The instances of a function block type are created in this application as an array of function blocks. The number of the array element index is displayed, i.e. the index of the function block linked to the visualisation. **This number can be changed in the visualisation** and permits the user to command the various function block instances and request their state data.
2	Reference index	The PLCopen specification declares a number of different data types which represent an axis or axis group etc. These data types act as a reference for the listed elements. The implemented PLC libraries define these references as an array of references (g_array_axis_ref, gAxesGroupRef) or the references are created for other data types as an array within the application. This number refers to the index of the element from an array of references created in the corresponding input/output variables for the selected FB. This value can only be edited for axis or axis group references when the global variables gP1AxisRefChgDisable, gP4AxesGrpRefChgDisable [▶ 11] are set to FALSE.
3	HLI index	This is a state datum and can therefore not be edited. The interface between the motion kernel and the PLC is organised as an array of structures which are assigned to axes or axis groups. The number is the index of the axis or axis-group-specific structure which is commanded by the selected function block.
4	Logical number	Identification number for axes and axis groups
5	All key	Press this key to set the Execute input pin to TRUE for all function block instances. This key is not provided in all visualisations.

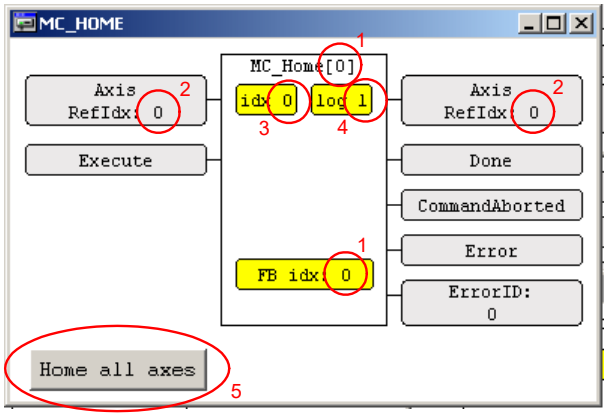


Fig. 3: Elements of a visualisation

4.15 Visualisation of axis positions

The MotionOfAxes visualisation is used to display the current positions of up to 10 axes. The positions are fetched from the HLI by the program of the same name and are saved in local variables which are then displayed by the visualisation. In addition, other state data is displayed for each axis and their meaning is described below.

Elements for visualisation of state data of an axis

State datum	Description
Ready	If green is indicated for Ready, it means that the drive is on standby and is controlled. Otherwise red is indicated.
Error	If green is indicated for Error, no error has occurred. The current axis position is displayed in the window next to the error state display.
Homed	Green is indicated when the axis is homed. Otherwise red is displayed
IsSAI	If green is indicated, this axis is supplied by a single axis interpolator and can be operated with modules according to Part 1 of the PLCopen specification. Red indicates that the axis was adopted in an axis group and can now be commanded by function blocks according to Part 4 of the PLCopen specification.
IsChAx	Green indicates that the axis was adopted in an axis group. The adjacent window displays the axis group number of the related axis group.

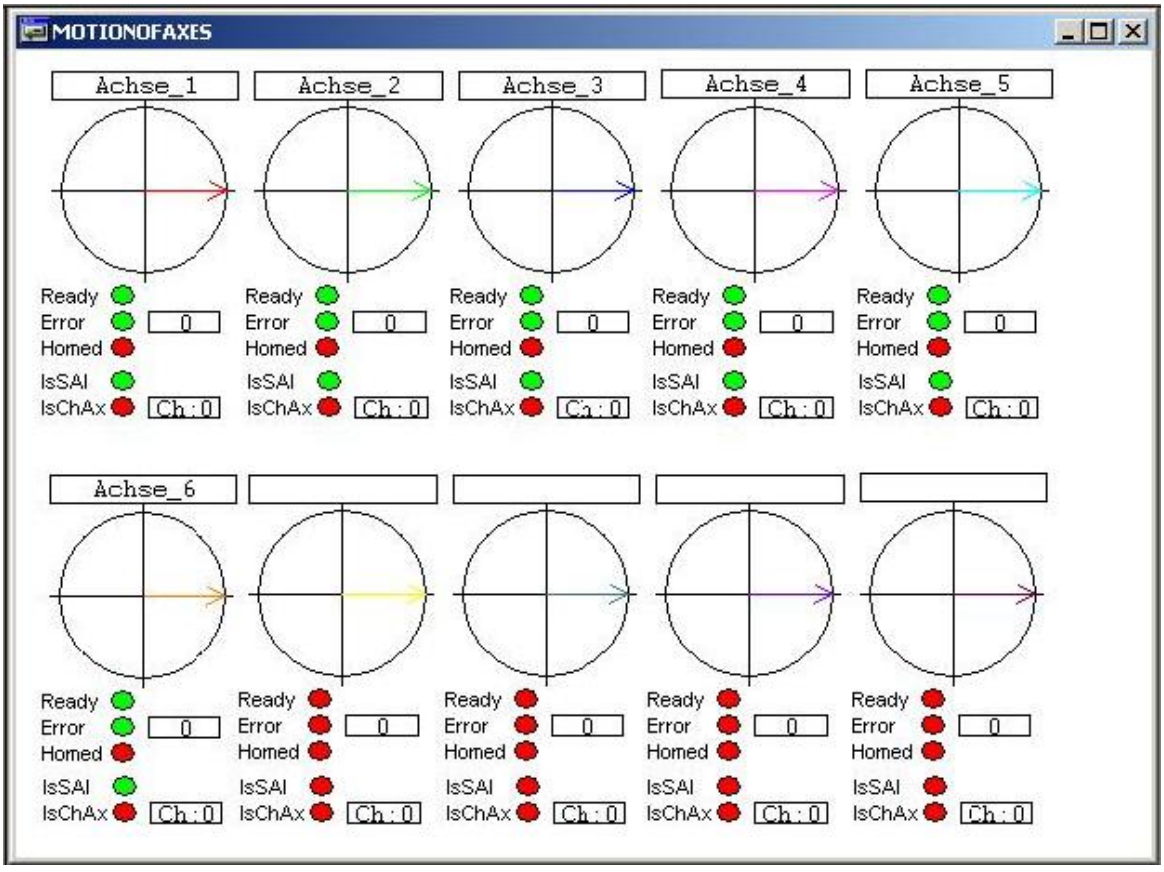
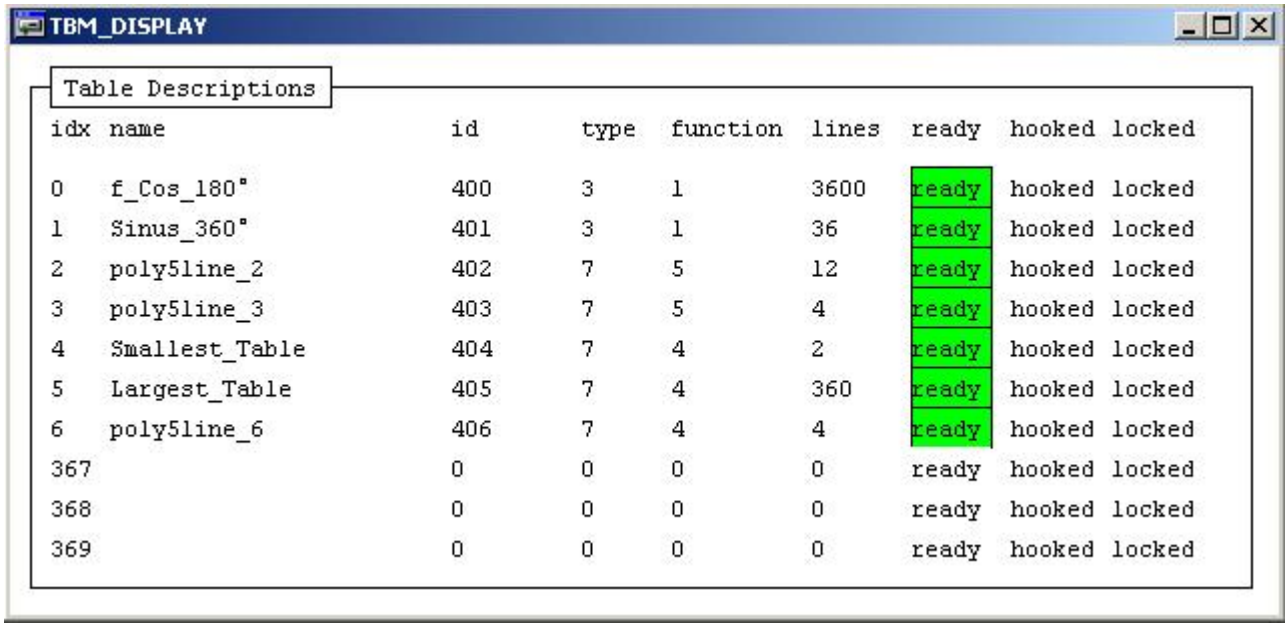
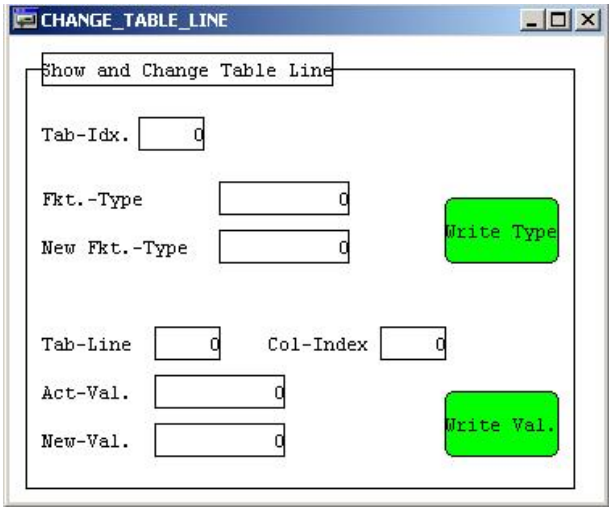


Fig. 4: MotionOfAxes visualisation

4.16 Visualisation of cam data



4.17 Visualisation to change cam tables



5 HLI BlockSearch – test application for the various block search variants

5.1 Introduction

The basic mechanisms implemented for the block search are contained in the functional description "Block search" (FCT-C6). The section below describes the commanding of the various block search variants by means of the PLC application HLI BlockSearch.

The PLC application is used to parameterise the block search and to observe the reaction of the NC kernel after starting the block search. The block search itself is activated when a block search type is selected in the PLC application, the required parameters are set and then a start of an NC program is requested. In the case of the example application, this is usually done via the **Online** tab of the corresponding channel in the SystemManager (TwinCAT 2) or the MOTION configuration (TwinCAT 3) by pressing the **Start** button there. For the purpose of simplification, the tab is referred to as Operation below

The block search is parameterised in the ADS program. There are no corresponding interfaces for this on the HLI. However, state data of the NC kernel can be retrieved via ADS and the HLI. State data can then be used to parameterise the block search via ADS communication. The handshake between PLC and NC kernel during the block search is processed by the corresponding control unit (CNC build >= 2800: **gpCh[i].bahn_lc_control.block_search**) on the HLI (see also [HLI//Block search], including diagrams on handshake).

The PLC application automatically acknowledges all technology functions (M, H, S, T) output by the NC kernel for channels 1 and 2 (see programme MAIN(), Fct: QuitChTechFunction()).

The PLC application can be used to parameterise the block search and then to select whether the handshakes required to execute the function are carried out automatically by the PLC application (see state machine at the end of POU BlockSearch) or whether the user wants to command them step-by-step via the visualisation.

Visualisations are structured so that

- the data in the column under the designation "**read command**" are the currently effective parameters in the NC kernel for the block search and were read out by the ADS.
- the data in the column under "**command to set**" are the parameters that will become active the next time a block search is commanded, provided they have been transferred to the NC kernel by ADS after being entered in the visualisation field by pressing the **set** button.
- the state data displayed in the column "**HLI act.value**" can be transferred to the corresponding "**command to set**" field on the HLI by pressing the **copy** button. This is used to adopt the data on the HLI, e.g. after a program abort, in order to parameterise the block search.

5.2 BlockSearch_Common visualisation

The upper section of this visualisation defines whether the required handshakes are automatically executed by the PLC application or whether the user commands the necessary handshakes by using the associated visualisation elements.

The panel highlighted in green is used to transfer the possible parameters to the NC kernel for all block search variants.

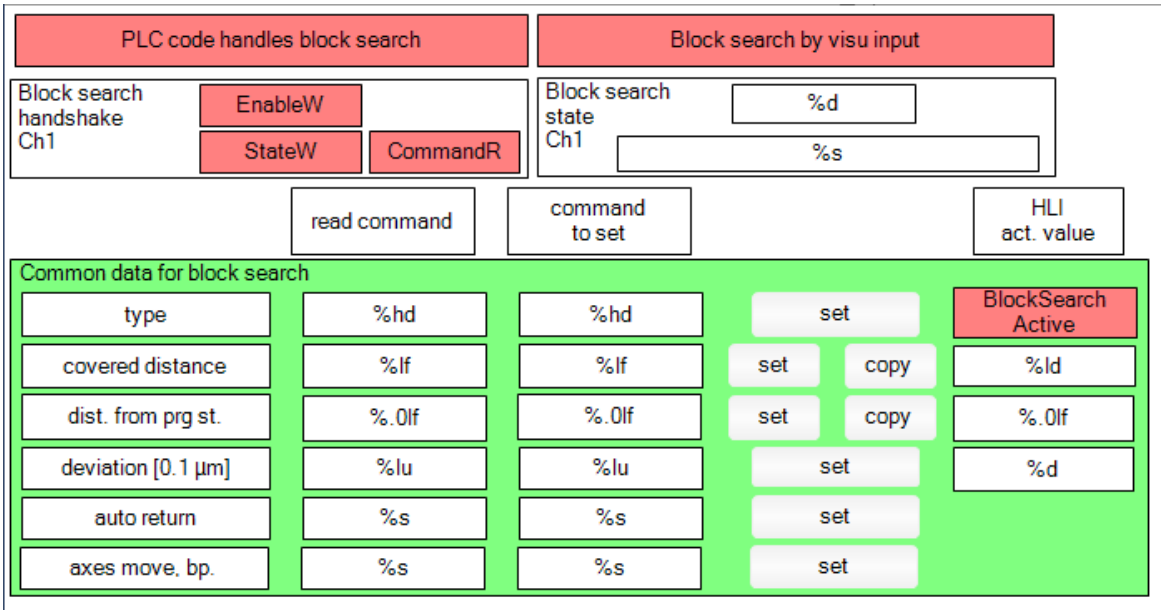


Fig. 5: BlockSearch_Common visualisation

5.2.1 Description of elements in the BlockSearch_Common visualisation

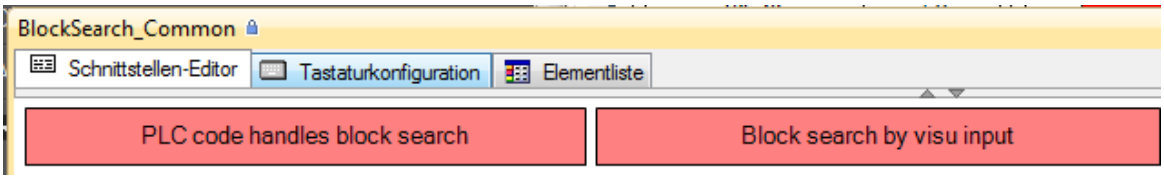


Fig. 6: Key: “PLC code handles block search" / "Block search by visu input”

Only one of the two keys can be active.

After the PLC application starts, the pre-assigned key **"Block search by visu input"** and the control unit are enabled on the HLI for the block search (enable_w of the control unit is TRUE). This is indicated by the green indicator "EnableW" in the **"Block search handshake CH1"** panel in the section below the two keys.

When **"PLC code handles block search"** is activated, all the handshakes required are processed by the state machine which is implemented at the end in the POU BlockSearch (program) after block search is parameterised and the NC program is started .

General block search parameters

Parameter	Description	Unit
Type	Block search type	
covered distance	Specifies the distance moved in an NC block	$^{0}/_{00}$
dist. from prg st.	Distance moved since program start or since the last occurrence of the NC command #DISTANCE PROG START CLEAR and activation of the calculation by #DISTANCE PROG START ON.	0.1 µm
Deviation	maximum deviation of the axes between actual position and continue position when processing is resumed after block search	0.1 µm
axes move bp	Specifies an automatic interrupt point by specifying the distance to program start	0.1 µm

5.2.2 Visualisations to assign specific block search parameters

The visualisations BlockSearch_Type_1, BlockSearch_Type_3 and BlockSearch_Type_4 are used to transfer the block search parameters to the NC kernel for the corresponding block search type.

For a description of block search types, see the functional description "Block search" (FCT-C6).

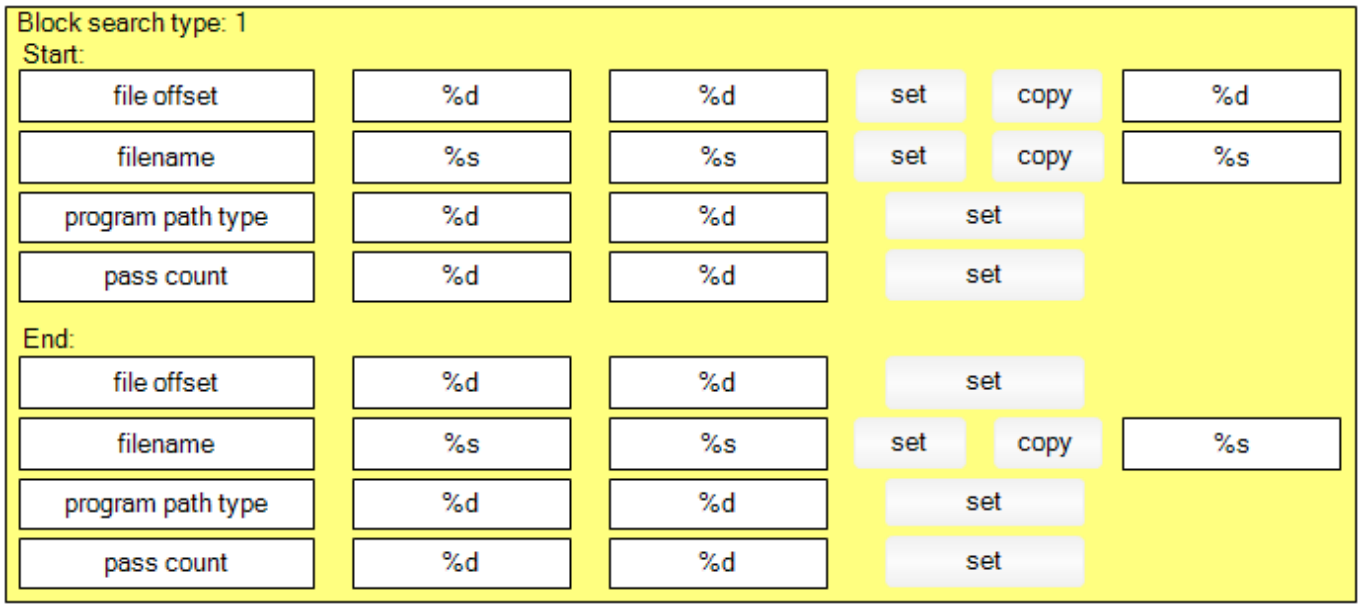


Fig. 7: Visualisation for parameterising block search type 1

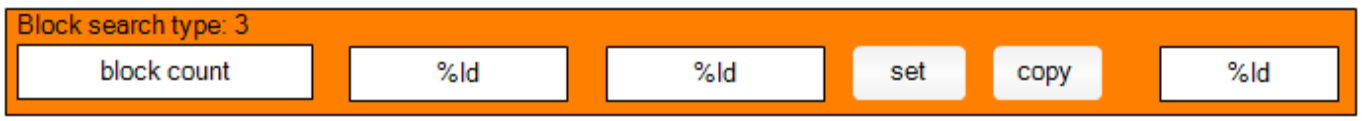


Fig. 8: Visualisation for parameterising block search type 3

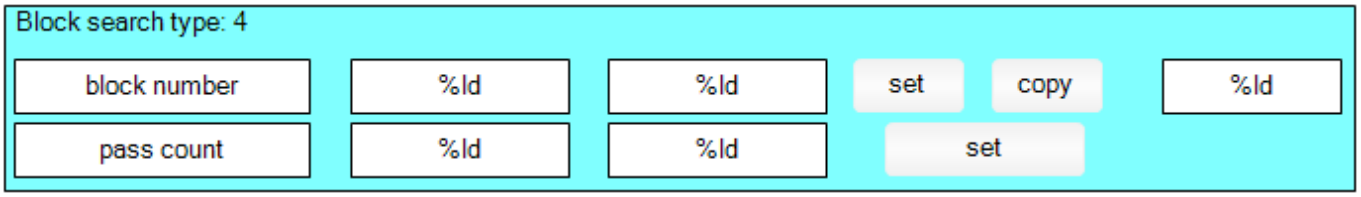


Fig. 9: Visualisation for parameterising block search type 4

5.3 Procedure with active "Block search by visu input" and active "auto return"

Notice

Start *X* means in this description that the start key of the SystemManager (TwinCAT 2) or the MOTION configuration (TwinCAT 3) is pressed in the online tab of the associated channel.

Start 1: Start NC program after parameterising block search

If a block search variant is selected by parameterisation, the NC kernel signals to the control unit after the start of the NC program that the block search is active by setting **command_r** . This is indicated by the button **"CommandR"** in the visualisation **BlockSearch_Common** when it turns green. The axes do not move during block search.

The block search state is displayed on the right of this in the **"Block search state Ch1"** section and is described by a text. The state comes from the element **gpCh[i].bahn_state.block_search_state_r** (CNC Build >= 2800) on the HLI. At this point in time the state is 1.

Handshake 1

The PLC must now confirm that it has received the message that block search is active and has completed its preparatory actions. It does this by setting the **state_w** of the control unit to TRUE. The user assumes this task for the PLC by clicking the button **"StateW"** which changes its colour to green to indicate that the **state_w** of the control unit was set to TRUE.

The block search then runs. The axes still do not move. State 2 is indicated as long as block search runs. When block search ends, state 3 is indicated and **command_r** in the control unit is set to FALSE. This is mirrored in the visualisation **BlockSearch_Common** so that the button **"CommandR"** returns to its original state – bright red.

Handshake 2

The PLC must then confirm that it has received the signal for block search end. For this reason, the user must click the button **"StateW"**. This sets **state_w** in the control unit to FALSE.

Block search then indicates state 4.

Start 2: Approaching the path

Provided that the block search parameter **"auto return"** was assigned the value 1, the axes approach on the **direct** path to the position specified in the block search parameters (continue position) and then stop.

State 5 is indicated as long as the axes are moving to the continue position. State 6 is indicated when the continue position is reached.

Start 3: Continuing the NC program

When the user presses the start key in the operation again, the NC program continues processing and the axes move accordingly.

5.3.1 Procedure with active "Block search by visu input" and deactivated "auto return"

NC program is started in the block search and **"auto return"** is parameterised as **deactivated**

Notice

If axes are moved after an NC program abort, ensure in **deactivated "auto return"** that the axes

1. are either located at the position (continue position) in the block search before start of the NC program. The continue position is specified by other block search parameters

2. or the value for the**"deviation"** is selected so that the current axis actual position results in a deviation from the set position which is less than the parameterised **"deviation"**.

If this is not the case, the error message 50474 – "Deviation of path after manual continue in the block search too large"(P-ERR-50474) is output.

Notice

The block search parameter specified by the user is calculated in the NC kernel after the block search. It is then processed in the NC program and the axes are then actually moved.

If this position is different from the real position resulting from the position of the machine axes, it means that the NC program moves the axes on the tool path with the corresponding offset at block search end.

The block search parameter **"deviation"** can define whether a deviation is permitted and the maximum amount it may assume. It must therefore be considered that, to permit the NC program to process with the existing offset after block search, it must be less than the parameterised **"deviation"**.

Attention

When **"auto return" is deactivated,** Start 2 does **not** execute the approach to the path (continue position). Instead the NC program is executed and the axes move – this therefore corresponds to Start 3.

5.3.2 Procedure with enabled "PLC code handles block search"

If the key **"PLC code handles block search"** appears in the visualisation, the user need not carry out any further interaction in the visualisation except for parameterising the block search type and then starting the NC program.

The required handshakes and starts are acknowledged and commanded by the automatic state indicator at the end of the POU BlockSearch.

Notice

As you can see from the automatic state indicator, Start 2 and Start 3 can also be commanded by the control unit gpCh[..]^.bahn_mc_control.continue_motion (CNC Build > 2800).

5.3.3 Other visualisations

5.3.3.1 AddFunctionality visualisation

The **AddFunctionality** visualisation enables the command of other functions related to block search and is used depending on the execution of an application.

For example

- the **"Start NC program at file offset"** section is used to start an NC program at the point which is defined by the specified file offset. The **set**- and **copy** keys are used for the same purpose as in other visualisations.
- the **"ContinueMotion CU"** section is used to command the appropriate control unit. This is useful if Start 2 and Start 3 are to be commanded by this control unit.
- the **"Forward/Backward"** key is used to execute the NC program backwards. This key acts on the NC kernel via the control unit **gpCh[..]^.bahn_mc_control.backward_motion**(CNC Build > 2800).
- the section **"Suspend axis output"** acts on the control unit **gpCh[..]^.bahn_mc_control.suspend_axis_output**(CNC Build > 2800). This can suppress the setpoint output of a channel to the physical axes and permit them to be requested and moved by another channel.

The command of the control unit is implemented in the POU "AddFunctionality".

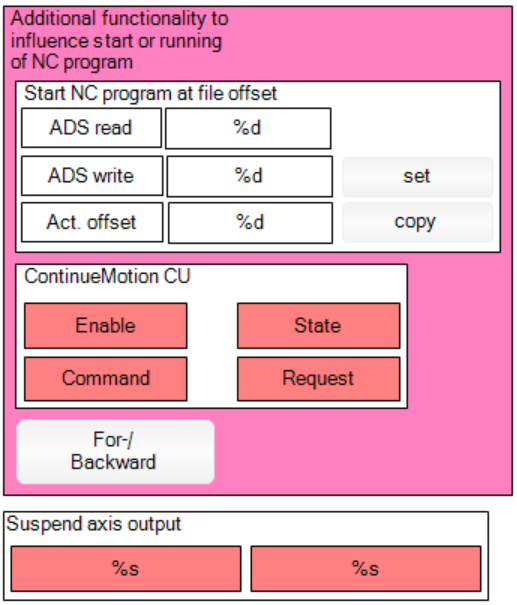


Fig. 10: Visualisation to command other functions

5.3.3.2 Ax_AxisCoupling visualisation

The control unit **gpAx[..]^.lr_mc_control.axis_coupling** (CNC Build > 2800) is used to command the axis couplings for an axis. It can affect the motion of one axis by the motion of other axes either additionally or exclusively.

This control unit is commanded in the POU ISG_FB_AXCU_AxisCoupling.

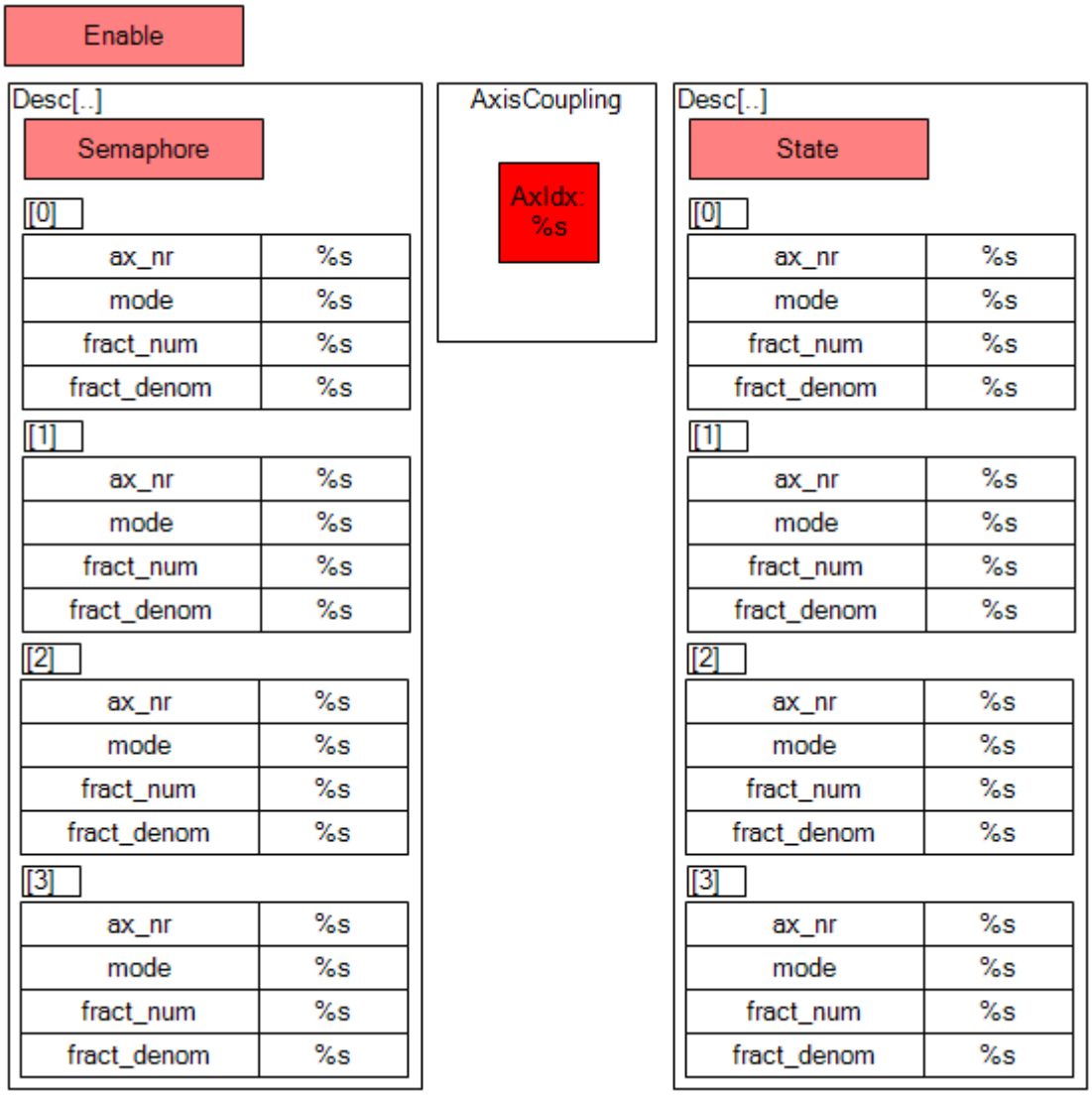


Fig. 11: Ax_AxisCoupling visualisation

5.3.4 Table of block search states

Number	Constant from HLI library	Description
0	HLI_BS_INACTIVE	Block search is not active
1	HLI_BS_WAIT_FOR_PLC_ON	NC kernel is waiting for the PLC to signal that it has detected that block search is active.
2	HLI_BS_ACTIVE	Block search is active. This is a dynamic state, which explains why no check should take place here.
3	HLI_BS_WAIT_FOR_PLC_OFF	NC kernel is waiting for the PLC to signal that it has detected that block search has ended.
4	HLI_BS_WAIT_RETURN_TO_CONTOUR	NC kernel is waiting for the signal to start the approach to the contour. This takes place either via the “Continue motion” control unit or via a program state (Automatic/ Active from PLC or start in the SystemManager).
5	HLI_BS_RETURNING_TO_CONTOUR	NC kernel moves the axes to approach the contour immediately.
6	HLI_BS_WAIT_FOR_CONTINUE_CONTOUR	The axes are moved so that the tool is located at the re-engagement position. Machining continues as from the re-engagement point after a command is received from the “Continue motion” control unit or after a program start in the SystemManager.

6 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).

7 Appendix

7.1 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

D

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de