MOOG

MSD PLC Manual

Description of
Basic Functionality and Libraries

Overview

MSD PLC

A PLC is a digital computer used for automation of electromechanical processes.

The MSD PLC can be used to extend the standard MSD functionality by

application specific functions.

Typical applications for using the MSD PLC may

- add new functions to the MSD Servo Drive (e.g. active position limitation)
- change existing functions in the MSD Servo Drive (e.g. a new homing function specific for an application)
- add new interface options (e.g. an analog input signal interface which is adding an offset to the position controller)
- do single axis camming and gearing

Audience

This manual will give a short introduction how to use the MSD PLC programming interface. With this manual the user can start creating own programs which can be downloaded to the MSD Servo Drive. This manual will give also an overview about the available functions in the MSD PLC libraries.

It is not the focus of this manual giving an overview how to use the programming environment CoDeSys 3.x. Please refer [COD] for an introduction to CoDeSys 3.x.

Technical alterations reserved

The contents of our documentation have been compiled with greatest care and in compliance with our present status of information.

Nevertheless we would like to point that this document cannot always be updated parallel to the technical further development of our products.
Information and specifications may be changed at any time. For information on the latest version please refer to drives-support@moog.com.

MSD PLC Manual

ID no.: CB15237-001, Rev. 1.4 10/2024

Applicable as from firmware version: V2.15

Table of Contents

Referenced Documents

Title	Document No. (English)	Document No. (German)
MSD Servo Drive Device Help	CB40859-001	CB40859-002
DRIVEADMINISTRATOR Manual	CA79186-001	CA79186-002
MSD Servo Drive Operation Manual	CA65642-001	CA65642-002
CoDeSys 3.x Installation and First Steps	In CoDeSys installation directory	

1　MSD PLC Characteristics

1.1　MSD-PLC Characteristics

For the MSD PLC no additional hardware is needed; the MSD PLC uses free CPU power of the MSD Servo Drive. No special MSD firmware is needed. The PLC is already included in the MSD Servo Drive standard firmware (2.15 or newer for MSD AC-AC **Servo Drives**). For the activation of the MSD PLC functionality a unique license code is needed.

Activation of the MSD PLC

If a MSD Servo Drive ordered with a MSD PLC license the MSD PLC functionality is already activated; no additional activation is needed.

For existing MSD Servo Drives without MSD PLC license the PLC functionality can be activated by entering the license at a later stage. This will be done using the Moog DRIVEADMINISTRATOR. A license code fits only to one unique drive.

Programming Environment

The creation of MSD PLC software will be done using the CoDeSys 3.x programming environment (IEC 61131 compliant). This programming environment can be downloaded at the CoDeSys website. Please contact the CoDeSys Support for details.

MSD PLC: Implemented Functions

- CoDeSys 3.x standard functions (e.g. timers, trigger-functions, flip-flops …)
- Functions for single-axis positioning, motor control, status and diagnosis
- Interpolated velocity and profile position mode; currently no cyclic modes are available
- CAM-table functions (MSD PLC CAM Table Library)
- Access to input- / output signals
- Further functions are planned: encoder simulation, watchdog functions

Performance

Number of Tasks	3 cyclic tasks (1 recommended), 3 event tasks, 3 free running tasks (1 recommended)
Minimal Cycle Time	The cycle time is limited by the available CPU power; cycle times down to 1 ms are possible - cycle times of 5 ms and more are recommended. The available CPU power depends on the activated functions in the MSD Servo Drive.
Program memory	512 kB flash memory
Data memory	512 kB SDRAM memory
Remanent variable memory	512 Byte retain (NVRAM) memory 512 Byte persistent (NVRAM) memory
Available ECAM	64
segments Input / Output Mapping	Direct access to 108 predefined parameters: analog and digital I/O and PLC-parameters (10 x floating point (REAL), 20 x integer (INT), 20 x double-integer (DINT))

1.2　Additional Documentation

During the installation of CoDeSys 3.x some documentation files will be copied into the CoDeSys directory (c:\Program Files\3S CoDeSys\CoDeSys\Documentation). The documentation is available in English and German language.

DocumentCoDeSys3.xInstallationandStart: information about the installation of CoDeSys 3.x and the creation of first CoDeSys programs.

CoDeSys 3.x Library description: description of the functions within the CoDeSys 3.x basic libraries (e.g. SysCom, SysFile, SysTime).

2 Getting started with the MSD PLC

2.1 Needed for using the MSD PLC

The MSD PLC can be used with MSD AC-AC Servo Drives (G392-xxxxxxxxxxx), MSD DC-AC Servo Drives (G393-xxx-xxxxxxx), MSD Compact Servo Drives (G394-xxx-xxxxxxx) and MSD liquid cooled Servo Drives (G395-xxx-xxxxxxx).

Needed for the use of the MSD PLC

- MSD Servo Drive with PLC license
- Moog DRIVEADMINISTRATOR" V5.3.5 or newer
- CoDeSys 3.x installation package
- MSD PLC drive description file (e.g. named "MSD.devdesc.xml")
- MSD PLC library files (e.g. named "MSDMCB.library")

CoDeSys, MSD PLC drive description and MSD PLC library files can be ordered at the Moog support.

2.1.1 MSD Servo Drive, ordered with a MSD PLC license

The MSD Servo Drive can be ordered together with a PLC license. These MSD Servo Drives are containing a "P" within the model number. Using a MSD AC-AC Servo Drive the ordering number may looks like: G392-xxx-xxxPxxx. For these drives the MSD PLC is activated automatically.

2.1.2 Single MSD PLC license

For existing MSD Servo Drives without MSD PLC license, this license can be ordered separately. With this code it is possible to activate the MSD PLC functionality. The ordering number for a single PLC-license is CA93105-001.

2.2 Installation of CoDeSys 3.x

Step	Action
1.	Unzip the CoDeSys 3.x files and store these in a local directory on your computer
2.	Double click onto the installation file
3.	Follow the instructions during the installation process

2.3 Entering License Code

If the MSD Servo Drive is already shipped with a MSD PLC license this step can be skipped.

Step	Action
1.	Open Moog DRIVEADMINISTRATOR (MDA)
2.	Enter PLC license code in parameter **P 0693** (Drive Settings -> Administration -> License)
3.	Store value in the non volatile memory and reboot the MSD Servo Drive

2.4 Installation of the MSD Device

The device description contains information which CoDeSys needs for getting access to the MSD Servo Drive. Without the device description communication between CoDeSys and MSD Servo Drive is not possible.

The device-description needs to be installed only once.

Step	Action
1.	Create a new "Devices" folder in the CoDeSys directory and create a subfolder for the MSD PLC device description (e.g. named V1.1.1.0_Moog)
2.	Copy the MSD PLC device description (3 files) to this folder. The MSD device description is available on the product CD or at the Moog support.
3.	Start CoDeSys, open device repository window (Tools / Device Repository)
4.	Browse to the device description directory (see above), click on "Install", choose and install MSD-device-description

2.5 Installation of MSD PLC Libraries

The MSD Servo Drive libraries are containing drive specific functions like functions to control the state machine, motion control functions, ECAM and EGEAR functions. The MSD-PLC libraries need to be installed only once.

Step	Action
1.	Copy MSD libraries to your hard disc. Recommended libraries directory: c:\Program Files\3S CoDeSys\Libraries. The MSD PLC library files can be found on the product CD or ordered at the Moog support.
2.	Start CoDeSys 3.x, Open the library repository window (Tools / Library repository), click on "install"
3.	Select file type "Library files" or "All files"
4.	Choose and install MSD-Motion-Control libraries from the directory containing the libraries (c:\Program Files\3S CoDeSys\Libraries)
5.	Close window

2.6 Creation and Basic Configuration of a new Project

2.6.1 Adding a new Project

A project contains all the information CoDeSys needs to create the software, which finally can be downloaded to the MSD Servo Drive. This includes program code, used libraries, task configuration, input- and output mapping and the communication setup.

Step	Action
1.	Click on "File" / "New Project"
2.	Click on "Empty Project"
3.	Change project name; click "OK"
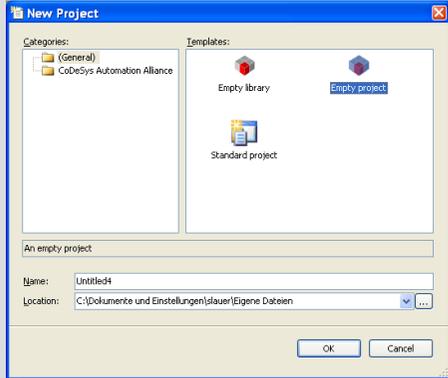
2.6.2   Adding a device to the project

Step	Action
1.	Right click on the white space in the project at the left
2.	Select "Add Object", "Device"
3.	Select the MSD Servo Drive
4.	Change the name of the device (e.g. into "MSD Sample")
5.	Click on "Add Device", click on "Close"

2.6.3   Adding a POU to the Project

A POU (Program Organisation Unit) contains the program code of the project.

Step	Action
1.	Right click on the "application" in the project tree at the left, select "Add Object", "POU"
2.	Enter a name for the POU (e.g. "MSD_PLC_Sample_Velocity"), choose type (program / function block / function) and implementation language. For details about POU types and implementation languages please refer [COD].
3.	Click on "Open"

The following example shows a POU for setting the MSD Servo Drive to "Operation Enabled". Used programming language in the picture below is CFC.
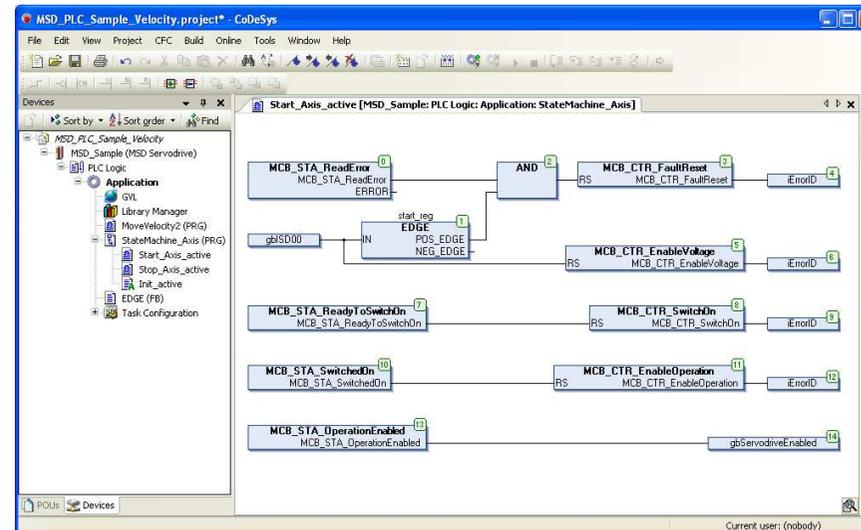
POUs can be created using different programming languages like CFC (graphical programming language), Structured Text, SFC (sequential function chart). Please refer [COD] for details.

2.6.4 Adding MSD Libraries to the Library Manager

For using drive specific functions one or more MSD PLC libraries need to be added to the project.

Step	Action
1.	Double click onto the "Library Manager" in the project tree on the left
2.	Choose "Add Library" out of the "Libraries" menu; select the library which should be added to the project
3.	Click on "OK"

2.6.5 Adding a Task Configuration and a Task to the Project

The task configuration contains one or more tasks of a project. Tasks will call

POUs frequently or on user defined events.

Step	Action
1.	Right click on the application in the project tree at the left, select "Add Object"

2.	Add a task configuration to the project
3.	Right click on the added task configuration, select "Add Object"
4.	Change the name of the task (e.g. into "Main_Task"), click "OK"
5.	Double click on the task in the tree on the left, configure the task by selecting priority, type and interval. Add a POU call to the task by clicking on "Add POU"
	Please note that tasks cycle times below 1 ms are not possible. Cycle times off 5 ms or higher are recommended. In case of a too low cycle time the MSD PLC will increase the cycle time automatically. A PLC task overflow message / error will be raised. The error reaction for this message can be set up using the Moog DRIVEADMINISTRATOR. Please refer MSD Servo Drive Device Help for details.

If the tab "Devices" at the bottom left corner of CoDeSys is selected, the project should look like the picture in the right.
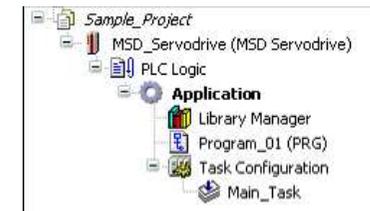
Event Tasks

Event tasks may be used to start functions in case of an event. An example may be an error handling mechanism which is triggered on the "Device_Error" event. For using an event driven task select "External" as task type and select the event.

The following list shows possible events for event triggered tasks.

Event Name	Description
Task_Overflow	Event posted continuously if PLC task overflow is
Init_Application	active Event posted after a reset, a download and an online change of the PLC application
Start_Application	Event posted after the actual application has been
Device_Error	started Event posted after an error occurs in the device
Device_Warning	Event posted after a warning occurs in the device
Start_Control	Event posted after start the control in the device
Stop_Control	Event posted after stop the control in the device
Input_xxx_RE	Event posted after a rising edge at digital input xxx
Input_xxx_FE	Event posted after a falling edge at digital input xxx
Output_xxx_RE	Event posted after a rising edge at digital output xxx
Output_xxx_FE	Event posted after a falling edge at digital output xxx
Int16_xxx_Bityyy_RE	Event posted after rising edge at bit yyy of PLC 16-bit integer variable xxx
Int16_xxx_Bityyy_FE 16-bit	Event posted after falling edge at bit yyy of PLC integer variable xxx

2.6.6   Set up of the I/O Mapping

The I/O mapping provides an easy and simple way to access some predefined parameters of the MSD Servo Drive. Accessing parameters using the I/O-mapping instead by using a function calls is much easier and faster.

In the left column of the I/O-mapping names can be given to parameters. A new global accessible variable with this name will be created.

Parameters available in the I/O-mapping are:

Parameter	Description	Read / Write
ENPO	Digital input at the MSD Servo Drive "enable power"	Read only
ISDSH	Digital input at the MSD Servo Drive Safe Stop (SH = "Safe Halt")	Read only
ISD00 - ISD05	Digital inputs at the MSD Servo Drive, accessible via connector X4	Read only
ISDA0 - ISDA1	Analog inputs at the MSD Servo Drive, accessible via connector X4	Read only
OSD00 -	Digital outputs at the MSD Servo Drive, accessible via connector X4	Read / Write
OSD03 REL1	Relays output (normally used for motor brake); please refer MSD Servo Drive Device Help for details	Read / Write
PARA_INT_0 - PARA_INT_19	20 16-bit integer parameters **P0850-P0859**, **P0880-P0889** (accessible via Moog DRIVEADMINISTRATOR)	Read / Write
PARA_DINT_0 - PARA_DINT_19	20 32-bit integer parameter **P 0860 - P 0869**, **P 0890-P 0899** (accessible via Moog DRIVEADMINISTRATOR)	Read / Write
PARA_REAL_0 - PARA_REAL_9	10 32-bit floating point / real parameter **P0870-P0879** (accessible via Moog DRIVEADMINISTRATOR)	Read / Write

i NOTE: For the use of digital outputs of the MSD Servo Drive it is needed to set the output-function (parameter **P 0122 - P 0124** and **P 0126**) to "PLC" (value 22) using the Moog DRIVEADMINISTRATOR.

i NOTE: Other parameters and values inside the MSD Servo Drive which are not available in the I/O-mapping can be accessed using parameter read / write functions. Please refer chapter 3.7 for details.

2.7 Communication setup

2.7.1 Set up of the communication

Step	Action
1.	Double click on "MSD_Sample" in the project-tree to open the PLC settings
2.	Add a new gateway (tab "communication settings")
3.	Click on "Scan Network" or add device manually (using device IP-address and Port 1740)
4.	Click on "Set active path"

2.7.2 Login into the MSD PLC

Step	Action
1.	Select Login from the "Online" menu
2.	Login with user name "root", password "root"

2.7.3 Creating a Boot Application

Boot applications will be stored in the non volatile memory. A boot application is

started automatically with the MSD Servo Drive (when 24V logic power is applied).

The boot-project can be created using the "create boot application" command from the "online"-menu.

2.8 Parameter Settings in the MSD Servo Drive

The MSD PLC is capable to control the MSD state machine (enabling / disabling the power stage, fault-reset) and the reference value of the drive (velocity or position). To hand over the drive control to the MSD PLC it is needed to set two parameters inside the MSD Servo Drive to "PLC Control". This can be done using the Moog DRIVEADMINISTRATOR.

Parameter Name	Parameter No.	Description
Source for MSD Drive Control Group: Motion profile / Basic settings	**P 0159 sub 0**	Control of the MSD State Machine; set parameter to "4" to enable control via PLC
Source for MSD Motion Reference Group: Motion profile / Basic settings	**P 0165 sub 0**	Control of the MSD Reference Command; set parameter to "4" (PLC basic library) to enable control via PLC

2.9 CPU Status

It is recommended to check the actual CPU load factor in the Moog DRIVEADMINISTRATOR (Drive status / Load factor) when the PLC program is running. The load factor shows the used and remaining computation power of the MSD Servo Drive CPU.

The load factor without a PLC program above should normally be around 70- 80 %. It is recommended to not exceed CPU load of 95% (when the MSD PLC is running). Above this value a cyclic PLC program call can not be guaranteed.

NOTE: The load factor depends on the PLC and also on the number and type of activated functions inside the MSD Servo Drive. Functions like field-weakening or gain, phase and offset correction (GPOC) are increasing the CPU load.

Id	Sub id	Name	Value	Unit	Introduction
1654	0	LU_Timing_Total	83,016	%	total computational load
1652		LU_Timing_Avg			SW Timing: Avgerage times
1651		LU_Timing_Max			SW Timing: Max. times
1653	0	LU_Timing_Reset	DONE		reset maximal and average timing values

3 MSD PLC Basic Motion Library

The MSD PLC Basic Motion Library offers functions which may be used to

- read / write parameters and variables from / to the MSD Servo Drive
- control the drive and getting information about the actual state
- read / write analog and digital outputs
- execute motion functions (e.g. positioning, velocity movements, …)
- execute homing functions

3.1 MSD Drive Control Functions

Using drive control functions the drive may be enabled / disabled. Also it is possible to clear errors and sending quick stop commands.

MCB_CTR_EnableOperation	This function changes the MSD Servo Drive state to "operation enabled" if "true"
MCB_CTR_EnableVoltag	This function changes the MSD Servo Drive state to "Enable voltage", if "true"
MCB_CTR_FaultReset	This function clears a drive error, if "true"
MCB_CTR_Power	This function changes the state to "operation enabled", including "enable voltage" and "switch on"
MCB_CTR_SwitchOn	This function enters the state "switched on"
MCB_CTR_QuickStop	This function is forcing a quickstop
MCB_CTR_Halt	This function commands a controlled quick stop

In the following, an example is given how the drive can be enabled using functions from the MSD PLC Basic Motion library.
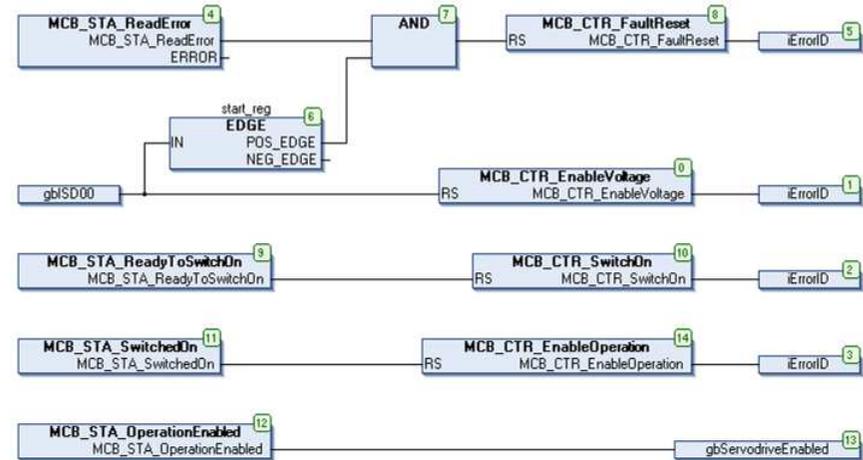
Parameter **P0159** needs to be set to 4 (PLC) for controlling the MSD Servo Drive state machine via MSD PLC. This parameter can be found in the Moog DRIVEADMINISTRATOR at Motion profile / Basic settings.

MOOG

3.2   Status Feedback Functions

Following functions are available for receiving a drive feedback. If a flag is returned the return value is Boolean.

MCB_STA_BrakeActiv	Returns the status of the break
MCB_STA_CurrentLim	Returns the current limitation flag
MCB_STA_DriveOperationEnabled	Returns the "Drive Operation Enabled" flag
MCB_STA_HaltActive	Returns the halt active flag
MCB_STA_HomingActive	Returns the homing active flag
MCB_STA_HomingAttained	Returns the homing attained flag
MCB_STAInternalLimit	Returns the internal limit flag
MCB_STA_JogModeActive	Returns the jog mode active flag
MCB_STA_MototrActive	Returns the motor active flag
MCB_STA_MotorStandstill	Returns the motor standstill flag
MCB_STS_NegativeRotation	Returns the negative rotation flag
MCB_STA_OperationEnabled	Returns the operation enabled flag
MCB_STA_Positive Rotation	Returns the positive rotation flag
MCB_STA_QiuckStop	Returns the quick stop flag
MCB_STA_ReadyToSwitchOn	Returns the ready to switch on flag
MCB_STA_SpeedLimit	Returns the speed limit flag
MCB_STA_SwitschedOn	Returns the switched on flag
MCB_STA_SwitchOnDisabled	Returns the switch on disabled flag
MCB_STA_TargetReached	Returns the target reached flag
MCB_STA_VoltageEnabled	Returns the voltage enabled flag
MCB_STA_Warning	Returns the warning flag

Please note that the feedback of the functions above depends on the parametrisation of the MSD Servo Drive.

MCB_STA_ReadError   Returns a possible error including the error ID (INT). Please refer the appendix for details
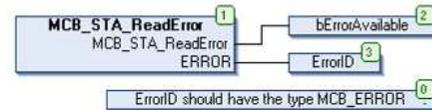
MCB_STA_StatWord   Returns the status word of the drive as an unsigned integer value (UINT). This status word contains the state of the drive and can be decoded as following:

MCB_STA_StatWord
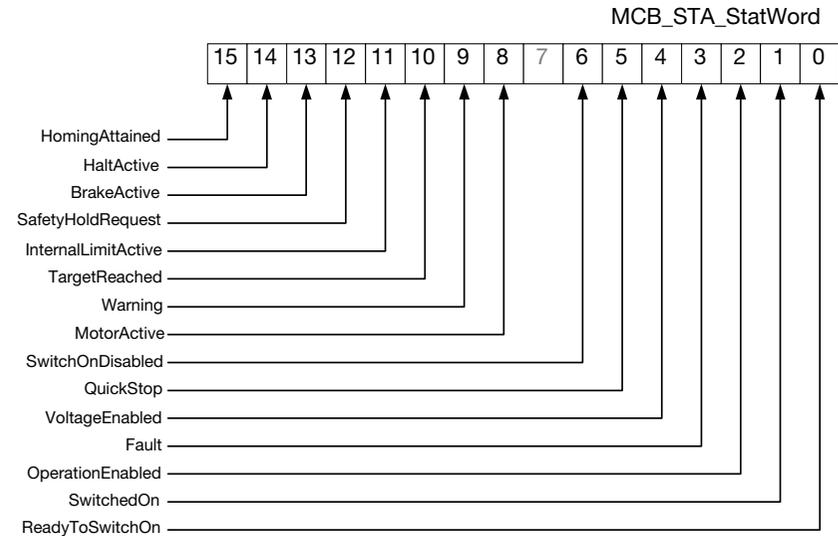
3.3   MSD Actual Value Functions

Several functions are available for reading actual values and the status of the

MSD Servo Drive.

MCB_ACT_Position   Returns the actual position in user units (DINT); please refer MSD Servo Drive Device Help for details how to work with user units.

MCB_ACT_RefPositio   Returns the reference position in user units (DINT)
MCB_ACT_ActVDC   Returns the actual DC link voltage in Volt (REAL)
MCB_ACT_RefVelocity   Returns the reference speed in RPM (REAL)
MCB_ACT_ActVelocity   Returns the actual speed in RPM (REAL)

3.4   Creation of application specific Errors

Using the function **MCB_CTR_SetError** the MSD Servo Drive can be set to error state.

The error-type is fixed to 31. The error sub-type ("location") can be defined using the function parameters. The Following example sets an error 31-07.

```
17    // Error 31-07 = Position difference monitoring with a too high position difference
18    IF bPositionErrorOcurred = TRUE THEN
19        MCB_CTR_SetError (Location := 07, Value := 07);
20    END_IF
```

This error will be shown on the drive display and in the DRIVEADMINISTRATOR.

Using the Moog DRIVEADMINISTRATOR it is possible to set the error reaction for PLC errors at parameter **P 0030 sup-ID 31**. This error reaction will be the same for all MSD PLC errors.

For details about MSD error reactions please refer the MSD Servo Drive Device Help.

3.5   MSD Motion functions

Following functions may be used to control the output of the drive in form of a torque, speed or position command.

Parameter **P 0165** needs to be set to 4 (PLC basic library) for enabling reference commands (speed / position) from the MSD PLC. This parameter can be found at Motion profile / Basic settings in the Moog DRIVEADMINISTRATOR.

3.5.1   MCB_CTR_Homing and MCB_Jog

MCB_CTR_Homing   "True" at the Boolean input "EXEC" starts a homing sequence; please refer the MSD Servo Drive Device Help for details about the homing functionality

MOOG

3.5.2   MCB_MoveAbsolute, Additive, Relative,

These functions are commanding a controlled motion to specified absolute position (**MoveAbsolute**), a relative distance to the last commanded position (**MoveAdditive**) or a relative position related to the actual position (**MoveRelative**).

EXEC	Executes the function if "true" (BOOL)
POS	Target position or relative position in user units (DINT)
VEL, ACC and	Maximum velocity (RPM), acceleration (RPM/s) and
DEC	deceleration (RPM/s). Data type: REAL
Return value	Possible Error code (INT) regarding list in the appendix

3.5.3   MCB_MoveStop

This function commands a controlled motion stop if input

EXEC	Executes the function if "true" (BOOL)
DEC	Value of deceleration (RPM/s) (REAL)
Return value	Possible Error code (INT) regarding list in the appendix

3.5.4   MCB_MoveVelocity

This function commands a continuous
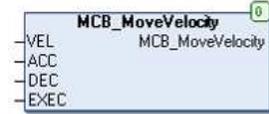
The inputs VEL, ACC and DEC are defining velocity (RPM), acceleration (RPM/s) and deceleration (RPM/s). The data type is (REAL).

3.5.5 MCB_MoveVelocityDirect,MCB_MoveTorqueDirect

These functions command a direct reference velocity / torque value.
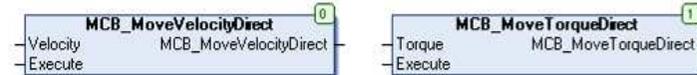
Execute	Executes the function if "true" (BOOL)
Velocity,	Target Velocity (RPM) / Torque (Nm). Data type: REAL
Torque Return	Possible Error code (INT) regarding list in the appendix
value	

For direct velocity and torque command it is needed to set the control mode to velocity (**MCB_MoveVelocityDirect**) or torque (**MCB_MoveTorqueDirect**). The move direct functions can only be executed in set-point profiling mode IP.

3.5.6 MCB_ResyncRefActPos

This function commands a resynchronisation of the reference position to the actual position and an additive position offset.

Execute   Executes the function if "true" (BOOL)
Offset   Target Position (DINT). in user units
ResetInteg   (BOOL) resets the integrative part of the control loop
Return value   Possible Error code (INT) regarding list in the appendix

The position offset will be added directly to the reference position without any ramp. A large position offset at the input "Offset" will result in an error reaction.

3.5.7 MCB_MoveAbsoluteDirect, MCB_MoveRelativeDirect
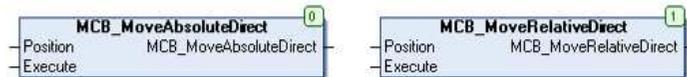
This function commands a direct motion to a specified absolute position / a position relative to the actual position. Both functions are only executable in position control mode "PCON" and with switched on set-point profiling mode IP.

Execute   Executes the function if "true" (BOOL)
Position   Target Position (DINT). in user units
Return value   Possible Error code (INT) regarding list in the appendix

3.6 MSD I/O Reading Functions

The following functions are returning the state of analog and digtal interface signals of the MSD Servo Drive. The motor brake output can be found on connector X12; all other signals are available at connector X4.

MCB_IO_ISA00   "true" if ISA00 > 2.4V, "false" if ISA00 < 0.4V (BOOL)
MCB_IO_ISA01   "true" if ISA01 > 2.4V, "false" if ISA01 < 0.4V (BOOL)
MCB_IO_ISA00_VAL   Value of Analog input 0, normalized to 1 (REAL) Value
MCB_IO_ISA01_VAL   of Analog input 1, normalized to 1 (REAL) Value of
MCB_IO_ISD0x   digital input x (BOOL)
MCB_IO_ISD0H   State of input "Safe Halt" (BOOL)
MCB_IO_MOTOR_BRAKE   State of motor brake (BOOL, "true" if active)
MCB_IO_OSD   State of digital output x (BOOL)
MCB_IO_ENPO   State of digital input "Enable Power Stage" (BOOL)
MCB_IO_REL   State of digital relay output "REL" (BOOL)
MCB_IO_RSH   State of digital output RSH (Safe Torque Off) (BOOL)

NOTE: The functions above can only be used for reading the state or value of the digital and analog interface signals. For writing output values it is recommended to use the I/O-Mapping functionality, described in chapter 2.6.6.

3.7 Parameter Access Functions

Following functions can be used for reading / writing parameters from / to the MSD Servo Drive.

The parameter itself needs to be addressed using parameter number and parameter sub-id. The Parameter number and sub-ID can be found in the MSD Servo Drive Device Help or using the Moog DRIVEADMINISTRATOR.

Example 1: Setting the value of paramter **P 0300 sub-ID 0** drive control mode) to 3 can be done using following function:

```
1   // write value to Parameter 300, Sub-ID 0
2   result = MCB_WriteUINTParameter (ID := 300, IDX := 0, VAL := 3);
3
```

Example 2: The value of **P 0300 sub-ID 0** can be read using following function:

```
1   // read value to Parameter 300, Sub-ID 0
2   result = MCB_ReadRealParameter (ID := 300, IDX := 0, VAL => return_value);
3
```

A return value different to zero will indicate a failure. Error codes are described in the appendix.

> Only parameters marked as "PDO capable" can be read / written in cyclic tasks using parameter access functions. Beside this all parameters / variables can be accessed in a free-wheeling task (background task). Please refer the MSD electronic datasheet (eds-file) for a list of "online capable" parameters.

Functions for reading parameter values	Functions to set parameter values
MCB_ReadBoolParameter	MCB_WriteBoolParameter
MCB_ReadDIntParameter (32-bit, signed)	MCB_WriteDIntParameter (32-bit, signed)
MCB_ReadIntParameter (16-bit, signed)	MCB_WriteIntParameter (16-bit, signed)
MCB_ReadRealParameter	MCB_WriteRealParameter
MCB_ReadSIntParameter (8-bit, signed)	MCB_WriteSIntParameter (8-bit, signed)
MCB_ReadUDIntParameter (32-bit, usign)	MCB_WriteUDIntParameter (32-bit, usign)
MCB_ReadUIntParameter (16-bit, usign)	MCB_WriteUIntParameter (16-bit, usign)
MCB_ReadUSIntParameter (8-bit, usign)	MCB_WriteUSIntParameter (8-bit, usign)
MCB_ReadStringParameter	MCB_WriteStringParameter

3.8 MSD: Additional functions

3.8.1 MCB_CTR_ForceCommutationDetection

This function forces new commutation detection at next power on.

Return value gives back an Possible Error code (INT) regarding list in the appendix.

3.8.2 Touch Probe functions

These functions are commanding a touch probe operation.

START	Starts the touch probe trigger if "true" (BOOL)
STOP	Stops the touch probe trigger if "true" (BOOL)
DONE	Gives back a "true" if finished (BOOL)
POS	Return Value: touchprobe position (DINT) in user units
Return value	Possible Error code (INT) regarding list in the appendix

The trigger should be set for one cycle only. A "true" at the output "DONE" indicates a new position. This position will be given at output "POS".

3.8.3 Pulse counter functions

These functions are counting pulses (both edges) on ISD05 (**PulseCountISD05**) or ISD06 (**PulseCountISD06**).

MCB_PulseDirectionCount realizes a pulse counter (both edges) at ISD05 with a direction input at IDS06.
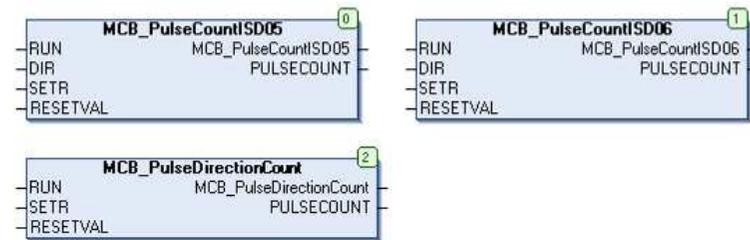
RUN	"true" starts the pulse counter (BOOL)
SETR	"true" sets the counter to zero (BOOL)
DIR	Direction "false" = negative, "true" = positive (BOOL)
PULSECOUNT	Returns the number of pulses (DINT)
Return value	Possible Error code (INT) regarding list in the appendix

4   MSD PLC CAM Table Library

The MSD PLC CAM Table Library offers functions which provide the MSD PLC CAM table functionality.

4.1   MSD Parameters for ECAM / EGEAR control

The ECAM / EGEAR parameter interface can be enabled via Parameter **P 0242 MPRO_ECAM_SyncModMode**. This parameter needs to be set to "EGEAR / ECAM" via PLC. The Moog DRIVEADMINISTRATOR may be used to set this parameter.

4.1.1   Master Encoder Configuration

The parameter for the internal master will be initialized together with system initialisation.

Parameters to configure the master encoder ECAM and EGEAR can be found in the Moog DRIVEADMINISTRATOR at Drive Settings / Motion profile:

Parameter name	Para-meter	Description
MPRO_ECAM_CAMMaster_AxisType	1319	Axis type of the master encoder
MPRO_ECAM_CAMMaster_RevLockMode	1320	Reverse lock mode of the internal master (calculated after the master gear)
MPRO_ECAM_CAMMaster_Amplitude	1321	Amplitude of the internal master calculation
MPRO_ECAM_CAMMaster_GearNum	1322	Numerator of the internal master gear
MPRO_ECAM_CAMMaster_GearDen	1323	Denominator of the internal master gear

Parameter name	Para-meter	Description
MPRO_ECAM_CAMMaster_SpeedFilTyp	1340	Type of the master encoder speed filter
MPRO_ECAM_CAMMaster_SpeedTFil	1327	Time constant of the selected master encoder speed filter
MPRO_ECAM_CAMMaster_SpeedFactor	1328	Speed factor for the internal master speed (calculated after the speed filter)
MPRO_ECAM_CAMMaster_Offset	1341	Position offset of the internal master

For using the parameter master (e.g. using field bus interface) following variables can be set:

Parameter name	Para-meter	Description
MPRO_ECAM_ParaMaster_ActPos	247	Actual position of the parameter master
MPRO_ECAM_ParaMaster_ActSpeed	248	Actual speed of the parameter master
MPRO_ECAM_ParaMaster_ActAcc	249	Actual acceleration of the parameter master
MPRO_ECAM_ParaMaster_Amplitude	250	Amplitude of the parameter master (2^n increments)

4.1.2 ECAM / EGEAR control word

Using the ECAM / EGEAR control word (parameter **P 1318** the ECAM / EGEAR functionality can be controlled.

This control word is bitwise coded. For executing a function the matching bit needs to be set to "true". This bit is reset automatically when the function is finished. It is also possible to use the corresponding function directly. Please refer the picture below for the corresponding function names.

MPRO_ECAM_ControlWord
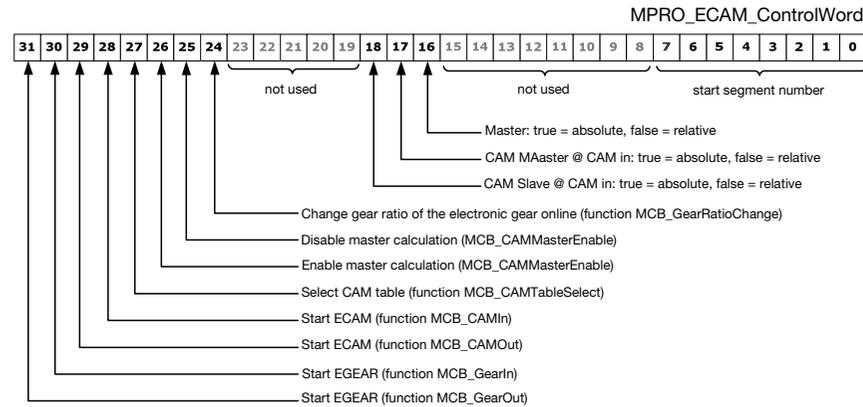
4.1.3 The ECAM / EGEAR Status Word

The **MPRO_ECAM_StatusWord** (parameter **P 1326** shows the actual ECAM and EGEAR state.

MPRO_ECAM_StatusWord
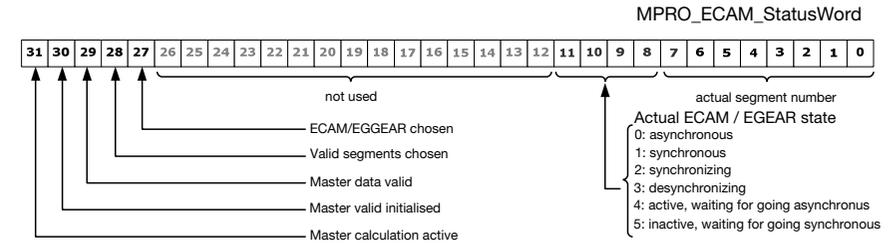
4.1.4 ECAM SegmentControlWord and SegmentStatusWord

Segment control word and Segment Status Word (parameters **P 1336 sub-ID 0-63** and **P 1337 sub-ID 0-63** in Drive Settings / Motion profile / Synchronized motion / Electronic camming / CAM table segments) are existing once for every segment.

This control word is bitwise coded. For executing a function the matching bit needs to be set to "true". This bit is reset automatically when the function is finished.

ECAM SegmentControlWord

ECAM SegmentStatusWord
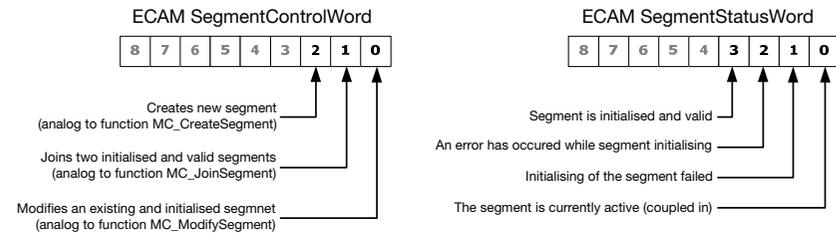
4.1.5 Using EGEAR with the parameter interface

The parameters for the EGEAR functionality can be entered manually or written via field bus.

Step	Action
1.	Set the synchronized motion mode to EGEAR using the Moog DRIVEADMINISTRATOR (parameter **P 0242** in Drive Settings / Motion profile / Synchronized motion)
2.	Set needed master data (Drive Settings / Motion profile / Synchronized motion / Master) Set needed EGEAR parameters (Drive Settings / Motion profile / Synchronized motion / Electronic Gearing)
3.	Start the drive, go to state "operation enabled"
4.	Enable the master calculation: set bit 26 in the ECAM/EGEAR control word
5.	Start the electronic gear: set bit 30 in the ECAM/EGEAR control word or use a digital input (rising edge to start ECAM/EGEAR) Stop the electronic gear: set bit 31 in the ECAM/EGEAR control word or use a digital input (falling edge to stop ECAM/EGEAR)
6.	Change the gear ratio online (direct change, no ramp): redefine the gear ratio (parameters **P 0251** and **P 0252**, set bit 24 in the ECAM/EGEAR control word

4.1.6 Virtual Master Parameter Interface

The virtual master can be used for a simulation of a real axis for testing the ECAM/EGEAR functionality or for encoder simulation (e.g. TLL-Out).

It is possible to simulate jerk-, acceleration- and deceleration-limited velocity ramp functions. A modulo value can be selected by the user. The Value will be limited automatically to next smaller 2^n value at the next initialisation of the virtual master.

All parameters (except reference speed) will be initialised at start of the virtual master. A change during operation has no effect (except reference speed). The reference speed may be changed during operation.

The virtual master can run also in "disabled" MSD Servo Drive state.

Parameters which may be set up for the virtual master:

Parameter name	Para-No.	Description
MPRO_ECAM_VM_Ctrl	267	Control / state variable of the virtual master
MPRO_ECAM_VM_Speed	244	Reference speed of the virtual master in RPM
MPRO_ECAM_VM_Amplitude	245	Amplitude of the virtual master in increments/revolution
MPRO_ECAM_VM_Acc	259	Acceleration of the virtual master in RPM/s
MPRO_ECAM_VM_Dec	260	Deceleration of the virtual master in RPM/s
MPRO_ECAM_VM_Jerk	261	Jerk of the virtual master in RPM/s^2
MPRO_ECAM_VM_PosAct	246	Actual modulo position of the virtual master in increments
MPRO_ECAM_VM_SpeedAct	262	Actual speed of the virtual master in RPM

Controlling the virtual master is possible using parameter **MPRO_ECAM_VM_Ctrl (P 0267)**.

Bit number	Description
-1	An error has occurred during the master initialisation
0	The virtual master is ready
1	Start the virtual master
2	Stop the virtual master (with ramp) Halt
3	the virtual master (without ramp)

4.1.7 Virtual Master Initialisation

The virtual master will be initialised, when the MSD Servo Drive switches to the state "operation enabled". If a system initialisation is triggered by parameter-changes then the virtual master will be stopped and initialized.

The virtual master will be stopped and initialized if one of the parameters are changed which are triggering a system initialisation.

Start / Stop / Halt: the virtual master can be started, writing a "1" to parameter **P 0267**. The virtual master starts with the user specified ramp. Writing a "2" will stop the virtual master with its user specific ramp. Writing a "3" to parameter **P 0267** the virtual master will stop immediately without a ramp.

Velocity change the virtual master reference velocity can be changed "online". The velocity will change with user specified ramp.

4.2   CAM Table Functions

4.2.1   MCB_CAMConfig

The CAM Table functions can be configured with the function **MCB_CAMConfig**.
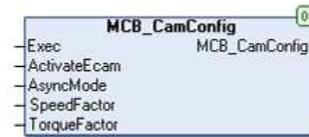
Exec	Executes the function if "true" (BOOL)
ActivateECAM	"true" activates the CAM functionality (BOOL)
AsyncMode	Control mode if CAM functionality gets asynchronous; data type: **MC_ASYNCMODE** (INT)
SpeedFactor	User specific speed factor (REAL, 1 will be 100%)
TorqueFactor	User specific speed factor (REAL, 1 will be 100%)
Return value	Error-ID (INT) regarding list in the appendix

If the input "ActivateECAM" is "false", the mode of synchronized motion will not be changed. To execute this function, the ECAM state needs to be asynchronous. In synchronous mode the user gets back an error "command not allowed".

CAM functionality can not be enabled in drive state "operation enabled".
Speed and torque factors are taking only effect on the feed forward values of the slave.

Data type MC_ASYNCMODE

This data type describes the drive internal interpolator mode which will be activated when the ECAM is changed to asynchronous mode.

Name	Value	Description
ASYNCMODE_SpeedMode	0	Ramp down from actual speed to zero speed after CAM out
ASYNCMODE_AbsPosMode	1	Absolute movement to the last ECAM reference position with the actual speed after CAM out; in this mode the axis will reach the last position with a small deviation which depends on the actual speed
ASYNCMODE_ContinueSpeedMode	2	Not implemented yet

4.2.2 MCB_CAMIn, MCB_CAMOut
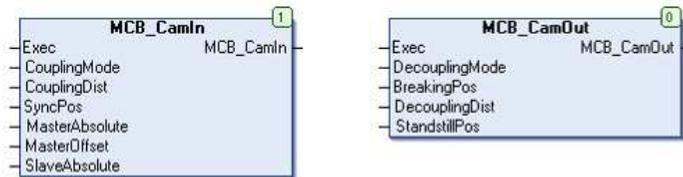
These functions engage / disengage the CAM functionality

Exec	Executes the function if "true" (BOOL)
CouplingMode	Coupling mode; (INT, data type **MC_CPLMOD**)
Decoupling Mode	Decoupling mode (INT, data type **MC_DECPLMOD**)
CouplingDist / DecouplingDist	Distance to couple in / out (DINT; user units)
Master / SlaveAbsolute	(BOOL) "true": master / slave reference absolute "false": master / slave reference relative
MasterOffset	Master offset
BreakingPos / StandstillPos	Breaking / standstill position concerning slave (DINT, user units)
DecouplingDist	Decoupling distance (DINT, user units)

Return value   Error-ID (INT) regarding list in the appendix

Notes:

- Select the start segment using the function **MCB_CAMTableSelect**
- **MCB_CAMMIn**: coupling position CP = SyncPos – CouplingDist, the coupling distance can be greater than the master amplitude
- **MCB_CAMOut**: coupling position DP = BreakingPos – DecouplingDist, the decoupling distance can be greater than the master amplitude
- An absolute master reference has no effect on changing the CAM table in asynchronous mode
- In case of absolute master reference, the master offset defines the master start position within the selected start segment
- The master offset must be less than the master distance of the selected start segment
- **MCB_CAMOut**: if desynchronizing is active (decoupling) and the decoupling mode is not "direct", a call of the function **MCB_CAMOut**
- **MCB_CAMOut**: if desynchronizing is active (decoupling) and the decoupling mode is "direct" a halt will be executed
- If synchronizing mode is active and **MCB_CAMIn** or **MCB_CAMOut** is called, a halt will be executed
- This functions will only be executed in device state "operation enabled"

Data type CPLMOD

Name	Value	Description
CPLMOD_Direct	0	CAM couples direct
CPLMOD_DirectPos	1	CAM couples direct when the master reaches the coupling position
CPLMOD_XFade	2	CAM couples in by weighting the curve and the start position with a normalized 5th order polynomial; the master distance to couple in is defined by the user; the start position fades out continuously and the curve position fades in continuously
CPLMOD_XFadePos	3	Same behaviour like CPLMOD_XFade, but the function starts in a way that the CAM is synchronous when the master reaches the synchronous position
CPLMOD_Linear, LinearPos	4, 5	Not implemented yet

Data type DECPLMOD - Decoupling Modes (desynchronize)

Name	Value	Description
DECPLMOD_Direct	0	CAM couples direct; user defined quick stop ramp will be used for deceleration
DECPLMOD_DirectPos	1	CAM couples direct when the master reaches the coupling position
DECPLMOD_DirectEndSeg	2	CAM couples direct at the end of the actual segment
DECPLMOD_DirectEndCyc	3	CAM couples direct at the end of the actual cycle
DECPLMOD_XFade	4	CAM couples out by weighting the curve and the standstill position with a normalized 5th order polynomial; the master distance to couple out is defined by the user; the curve position fades out continuously and the standstill position of the slave fades in continuously
DECPLMOD_XFadePos	5	Comparable to CPLMOD_XFade; CAM is asynchronous when the master reaches the decoupling position
DECPLMOD_XFadeEndSeg	6	Comparable to CPLMOD_XFade; decoupling starts, when the master reaches the end of the actual segment
DECPLMOD_XFadeEndCyc	7	Comparable to CPLMOD_XFade; decoupling starts, when the master reaches the end of the actual cycle
DECPLMOD_Linear, DECPLMOD_LinearPos, DECPLMOD_LinearEndSeg,	8 9 10 11	Not implemented yet
DECPLMOD_LinearEndCyc DECPLMOD_FadeOut	12	Same like CPLMOD_XFade, without standstill position
DECPLMOD_FadeOutPos	13	Same like CPLMOD_XFadePos, without standstill position
DECPLMOD_FadeOutEndSeg	14	Same like CPLMOD_XFadeEndSeg, without standstill position
DECPLMOD_FadeOutEndCyc	15	Same like CPLMOD_XFadeEndCyc, without standstill position

4.2.3 MCB_CAMTableSelect

This function selects the CAM tables by setting the connections to the relevant segment.
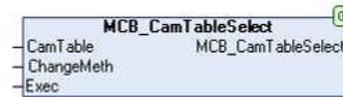
Exec	Executes the function if "true" (BOOL)
CAMTable	Reference to the CAM segment (UINT)
ChangeMeth	Change method of the CAM tables (segment) (INT)
Return value	Error-ID (INT) regarding list in the appendix

Notes:

- If this function is called in decoupled mode, the start segment will be set; in coupled mode the CAM table will be changed "online".
- If a CAM table change is already active, the actual change CAM job will be stopped and the new job will be assigned.
- To select a CAM table, choose its first segment.

Data type MC_CCMETH

Methods to change the CAM table in the synchronised state.

Name	Value	Description
CCMETH_Locked	0	No change of the CAM table is allowed
CCMETH_Break	1	The CAM table changes immediately
CCMETH_SwitchSeg	2	The CAM table changes at the end of the actual segment
CCMETH_SwitchCyc	3	The CAM table changes when an end of cycle signal occurs (see MC_SEGMOD)

4.3 Gearing Functions (MSD CAM Library)

4.3.1 MCB_GearConfig
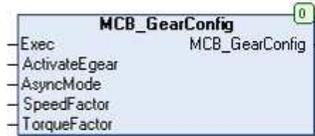
Exec	Executes the function if "true" (BOOL)
ActivateEGEAR	Activates the EGEAR functionality, if "true" (BOOL)
AsyncMode	Control mode when the CAM gets asynchronous; data type: **MC_ASYNCMODE** (INT)
Speed factor / torque factor	User specific speed / torque factor (1 = 100%) (REAL)
Return value	Error ID (INT) regarding list in the appendix

Notes:

- Torque and speed factors take only effect on the feed forward torque / speed of the slave.
- If input "ActivateEGEAR" is "false", the mode of synchronized motion will not be changed.
- To execute this function, the EGEAR state needs to be asynchronous.
- The EGEAR can not be activated in the MSD Servo Drive state "operation enabled".

4.3.2 MCB_GearIn

This function commands a ratio between the position / velocity of the slave and master axis.
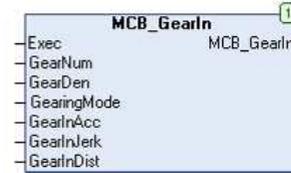
Exec	Executes the function if "true" (BOOL)
GearNum / GearDen	Gear numerator / denominator (DINT / UDINT)
GearingMode	Mode to gear in; data type **MC_GEARMOD** (INT)
GearInAcc	Acceleration of the ramp to gear in (REAL; RPM/s)
GearInJerk	Jerk in the ramp to gear in (REAL; RPM/s)
GearInDist	Gear in distance (fade, crossfade) (UDINT; user units)
Return value	Error-ID (INT) regarding list in the appendix

Notes:

- The parameters GearInAcc and GearInJerk are used by the gear in mode **GEARMOD_Ramp**.
- GearInDist is only used by the gear in modes **GEARMOD_Fade** / **GEARMOD_CrossFade**.
- This function can be assigned every time, but it will be only execute when the device is in the state "operation enabled".

Data type MC_GEARMOD

The behaviour of the modes is analog to the ECAM coupling modes.

Name	Value	Coupling Mode Description
GEARMOD_Direct	0	Gearing direct
GEARMOD_Ramp	1	Gearing via velocity ramp
GEARMOD_Fade	2	Coupling via fading
GEARMOD_XFade	3	Coupling via crossfade function

In case of direct coupling mode, the EGEAR functionality will forward its reference generating to the drive internal interpolator. In this case the drive will decelerate by the user defined quick stop ramp. If the drive should stop immediately, the quick stop deceleration (parameter **MPRO_402_QuickStopDec, P 2242** needs to be zero.

Note: if the coupling mode crossfade is chosen, it may occur that the speed overshoot is larger than the maximum motor speed. In this case the reference speed can't be reached.

4.3.3   MCB_GearOut

This function disengages the slave axis from the master axis.
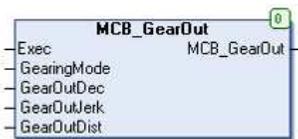
Exec	Executes the function if "true" (BOOL)
GearingMode	Mode to gear out (INT, Data type: **MC_GEARMOD**)
GearOutDec	Deceleration to gear out (REAL, RPM/s)
GearOutDist	Distance to gear out (only used in **GEARMOD_Fade**)
GearOutAcc / GearOutJerk	Acceleration and jerk values (REAL, RPM/s and RPM/s²); only used in **GEARMOD_Ramp**
Return value	Online-Shops der Watchlist InternetError-ID

(INT) regarding list in appendix
Notes:

- **MCB_GearOut** will be ignored if desynchronizing is active and the gearing mode is not direct.
- If synchronizing is active a halt will be executed.
- If desynchronizing is active and the decoupling mode is direct a halt will be executed.
- If this function is called with an unknown gearing mode, the EGEAR will couple out directly.
- If the mode to gear out is set to **GEARMOD_Ramp** and the deceleration is zero, the EGEAR will couple out directly.
- This function will only be executed in drive state "operation enabled".

4.3.4   MCB_GearOut (Absolute, Relative, Velocity)

MCB_MoveAbsoluteGearOut disengages the slave axis from the master axis and commands a controlled motion to specified absolute position without stop.

MCB_MoveRelativeGearOut disengages the slave axis from the master axis and commands a controlled motion of a specified distance relative to the actual position at the time of execution without stop.

MCB_MoveVelocityGearOut disengages the slave axis from the master axis and commands a continuous velocity without stop.
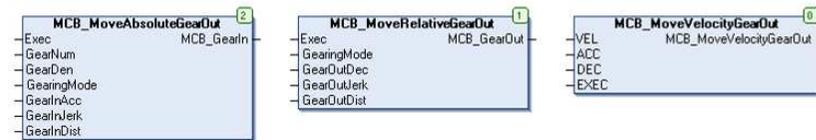
Exec	Executes the function if "true" (BOOL)
Pos	Target position (DINT in user units)
Vel	Maximum velocity (REAL in RPM)
ACC / DEC	Acceleration and deceleration (REAL in RPM/s)
Return value	Error-ID (INT) regarding list in the appendix

4.3.5   MCB_GearRatioChange

This function changes the gear ratio of the electronic gearing online.

Exec	Executes the function if "true" (BOOL)
GearNum	Gear numerator (DINT)
GearDen	Gear denominator (UDINT)
Return value	Error-ID (INT) regarding list in the appendix

Note: in synchronous mode this function will only have effect on the electronic gear ratio.

CAM Table Library

4.4 CAM Table Master Functions

4.4.1 MCB_CAMMasterEnable

Exec   "true" activates the CAM master (BOOL)
Return value   Error-ID (INT) regarding list in the appendix

• To execute this function, the function **MCB_CAMMasterSelect** needs to be called first.
• This function will be assigned every time, but will only be executed in "operation enabled" state.

4.4.2 MCB_CAMMasterSelect

This function selects and configures the CAM master. This function needs to be called once before switching the MSD Servo Drive state to "operation enabled". If this function is called while the master is active, the master will be deactivated.
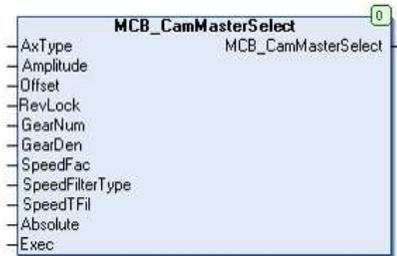
The amplitude of the master encoder will be aligned to the amplitude of the ECAM/EGEAR master axis. If the master gear ratio is 1:1 then one revolution of the CAM master represents one revolution of the physical master encoder axis.

The master position is not calculated while operation is disabled.

Function Parameters:

Exec	Executes the function if "true"
AxType	Axis type of the master reference, data type **MC_AXISTYPE**
RevLock	Reverse lock mode; Data type **MC_REVLOCK** The reverse lock modes are active in "engaged" and "disengaged" state
GearNum / GearDen	Gear numerator [-32767 - 32768] and gear denominator [1 - 65536]; by changing the gear ratio of the CAM master, the relationship between the physical master and the CAM master can be changed
SpeedFac	Speed factor of the master velocity [-1000 - 1000]; the speed factor can be used to increase or decrease the calculated master velocity
SpeedFilterType	Filter type of the master speed; data type: **MC_MASTERSPDFILTYPE**
SpeedTFil	Time constant of the master velocity filter: 0 - 100 ms
Amplitude	Amplitude of the ECAM/EGEAR master axis
Offset	Offset of the master position; the master offset will added to the start position
Absolute	Master relationship absolute; at <u>absolute master relationship</u>, the single turn part of the master position will be synchronized to the master reference position including the master offset; at <u>relative master relationship</u>, the master position starts the master offset
Return value	Error-ID (INT) regarding list in the appendix

Data type MC_AXISTYPE

Name	Value	Description
AXISTYPE_NoAxis	0	No axis
AXISTYPE_Virtual	1	Virtual master
AXISTYPE_Para	2	Parameter interface master
AXISTYPE_EncCh1	3	Encoder channel 1 X7 (SinCos)
AXISTYPE_EncCh2	4	Encoder channel 2 X6 (Resolver)
AXISTYPE_EncCh3	5	Encoder channel 3 X8 (Optional)

Data type MC_RECLOCK

Name	Value	Description
REVLOCK_Inactive	0	Reverse lock inactive
REVLOCK_ActiveComp	1	Reverse lock active - with way compensation
REVLOCK_Active	2	Reverse lock active - without way compensation

Data type MC_MASTERSPDFILTYPE

Name	Value	Description
MASTERSPDFILTYPE_Off	0	No filter
MASTERSPDFILTYPE_PT1	1	PT1 filter
MASTERSPDFILTYPE_Avg	2	Average filter

4.4.3   MCB_VirtualMasterSetPara

This function sets the parameter set of the virtual master. The master will be initialised when the drive goes to MSD Servo Drive state "operation enabled".

This function can not be called in operation enabled state. The function needs to be called before **MCB_CAMMasterSelect**. If this function is called while the virtual master is active, the virtual master will be deactivated at the next initialisation of the virtual master.

The virtual master can be used as source for the encoder simulation.

Exec	Executes the activates CAM master (BOOL)
Amplitude	Virtual master amplitude in increments; the Amplitude will be rounded down to the nearest 2^n value (UDINT)

Velocity	Velocity in RPM (REAL); the velocity of the virtual master can be changed online by the function **MCB_VirtualMasterSetVel** (see below)
Acceleration / Deceleration	Maximum acceleration / deceleration in RPM/s (REAL)
Jerk	Value of the jerk in RPM/s^2 (REAL)
Return value	Error-ID (INT) regarding list in the appendix

4.4.4   MCB_VirtualMasterSetVel

This function sets the velocity of the virtual master.

Exec	Executes the function
Return value	Error-ID (INT) regarding list in the appendix

4.4.5   MCB_VirtualMasterStart / -Stop / -Halt

MCB_VirtualMasterStart starts the virtual master.
MCB_VirtualMasterStop stops the virtual master using a ramp.
MCB_VirtualMasterHalt stops the virtual master directly without a ramp.

Before the virtual master can be started, the function **MCB_VirtualMasterSetPara** needs to be called once at least.

Exec	"true" activates the function (BOOL) Error-ID
Return value	(INT) regarding list in the appendix

5 MSD PLC CAM tools library

The MSD PLC CAM tools library offers tools for the creation and modification of CAM-Table segments. This library includes automatically the standard library.

5.1 Master modification and information

5.1.1 MC_GetMasterPos / MC_GetMasterVel

These functions return the actual master position / the master velocity.

MC_GetMasterPos	Absolute master position (DINT, user units)
SingleTurn MultiTurn	Single turn part of the master position (DINT)
Actual master velocity	Multi turn part of the master position (DINT)
	Actual master velocity (RPM, Data type: REAL)

5.1.2 MC_AddMasterPos

This function adds an incremental position direct to the master position.

This function can be called online, so the actual master position will changed directly. In synchronous state this function can generate massive jumps in the reference value. If this function is called in asynchronous state, the assigned additive master position will be overwritten and added at the first synchronous cycle.

AddPos	Additive master position (DINT)
Return value	Error-ID (INT) regarding list in the appendix

5.1.3 MC_SetMasterPosST / MC_SetMasterPosMT

MC_SetMasterPosST	Sets the actual master position single turn part (DINT)
MC_SetMasterPosMT	Sets the actual master position multi turn part (DINT)
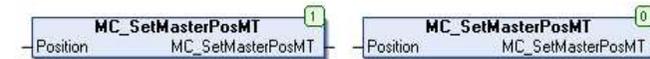
Position	New single / multi turn part of master position (UDINT)
Return value	Error-ID (INT) regarding list in the appendix

Note: the master position can not be changed online.

5.1.4 MC_ResetMasterPos

This function sets the master position to zero (single-turn and multi-turn information). The master position can not be set to zero in the drive state "operation enabled".

Return value: Error-ID (INT) regarding list in the appendix

5.1.5 MC_CAMMasterEnabled

This function returns the actual state of the CAM master.
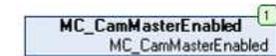
Return value	Returns "true" if the CAM master is enabled (BOOL)

5.1.6 MC_VirtualMasterActive

Return value gives back "true" if the virtual master is active (BOOL)

5.2 CAM Table functions: Segment Creation and Modification

5.2.1 MC_ChangeCAMActive

Return value gives back "true" camming is active (BOOL)

5.2.2 MC_CycleCounter

This function counts the passed ECAM cycles.

Reset	"true" sets the counters to zero (BOOL)
CyclesPos	increases, if a segment with the attribute **SEGMOD_Pos** left in positive direction (UDINT)
CyclesNeg	increases, if a segment with the attribute **SEGMOD_Neg** left in negative direction (UDINT)
Return value	Error-ID (INT) regarding list in the appendix

Notes:

- The end of an ECAM cycle is defined by the modes of the segments (see **MC_CreateSegment**).
- If the axis stands still between two cycles and the actual cycle toggles between two segments, the counters increases while every segment change.

5.2.3 MC_CycleSlavePos

This function calculates the absolute slave position corresponding to a specific master position and an associated start segment.
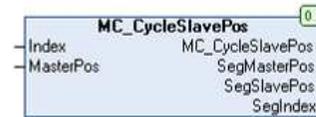
Index	Start index of the calculation (UINT)
MasterPos	Master position for calculation (DINT)
SegMasterPos	Master position in the calculated segment (UDINT)
SegSlavePos	Calculated slave position (DINT)
SegIndex	Calculated segment (UINT)

Depending on the chosen master position, the calculation of the return values could take a long time. This function should be used in freewheeling PLC cycle only.

5.2.4 MC_GetECAMState

This function returns the state of the ECAM.

Return value	State of ECAM (INT, data type **MC_ECAMSTATE**

Data type MC_ECAMSTATE

Name	Value	Description
ECAMSTATE_Async	0	ECAM is asynchronous
ECAMSTATE_Sync	1	ECAM is synchronous
ECAMSTATE_Synchronising	2	ECAM is synchronizing
ECAMSTATE_Desynchronising	3	ECAM is desynchronizing

5.2.5   MC_CreateSegment

This function **MC_CreateSegment** creates a new segment in the CAM table. This function sets the data of a segment without boundary conditions of the next and the previous segment. This function can not be called in state "operation enabled". These new segments will be initialized at switching to "operation enabled".
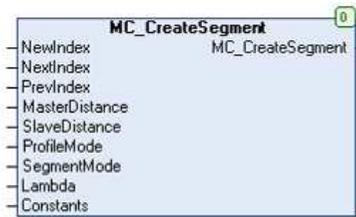
NewIndex NextIndex / PrevIndex	New index of the segment (UINT) Following / previous index of this segment (UINT)
MasterDistance	Distance of the master of this segment (UINT; increments)
SlaveDistance	Distance of the slave of this segment (UINT; increments)
ProfileMode	Segment profile mode; data type **MC_PROFILES (**INT)
SegmentMode	Segment: mode; data type **MC_SEGMOD** (INT)
Lambda	Inflexion point parameter; set lambda to 0.5 for a symmetrical profile (REAL)
Constants	Possibly needed constants to define the segment, depending on the profile mode (ARRAY [1..10] of REAL)
Return value	Error-ID (INT) regarding list in the appendix

Data type MC_PROFILES

(Value), Name	Value	Description
PROF_Inactive	0	Inactive segment – can't be used as active ECAM segment
PROF_Stop	1	Stop segment (CAM profile is immediately disengaged)
PROF_Standstill	2	Standstill segment
PROF_V_V_Linear	3	Velocity to velocity straight line
PROF_R_R_SinSimp	4	Rest to rest simple sine
PROF_R_R_Poly5	5	Rest to rest polynomial 5th order
PROF_R_R_ModAccTrap	6	Rest to rest modified acceleration trapezoid
PROF_R_R_SlopingSinus	7	Rest to rest sloping sinus
PROF_R_R_ModSinus	8	Rest to rest modified sine
PROF_R_V_ModSinus	9	Rest to velocity modified sine (constants[1] = normalized end velocity)
PROF_R_V_ModSinusEq AccDec	10	Rest to velocity modified sine (deceleration = acceleration) (constants[1] = normalized end velocity)
PROF_R_V_Poly5	11	Rest to velocity polynomial 5th order sine (constants[1] = normalized end velocity)
PROF_R_V_Poly5_Norm	12	Rest to velocity polynomial 5th order with normalized end velocity
PROF_V_R_ModSinus	13	Velocity to rest modified sine (constants[1] = normalized start velocity)
PROF_V_R_ModSinusEq AccDec	14	Velocity to rest modified sine (deceleration = acceleration) (constants[1] = normalized start velocity)
PROF_V_R_Poly5	15	Velocity to rest polynomial 5th order (constants[1] = normalized start velocity)
PROF_V_R_Poly5_Norm	16	Velocity to rest polynomial 5th order with normalized start velocity
PROF_V_V_ModSinus	17	Velocity to velocity modified sine (constants[1] = normalized start velocity, constants[2] = normalized end velocity)
PROF_V_V_ModSinusEq AccDec	18	Velocity to velocity modified sine (deceleration = acceleration) (constants[1] = normalized start velocity, constants[2] = normalized end velocity)

PROF_V_V_Poly5	19	Velocity to velocity polynomial 5th order (constants[1] = normalized start velocity, constants[2] = normalized end velocity)
PROF_M_M_Poly5	20	Motion to motion polynomial 5th order (constants[1] = normalized start velocity, constants[2] = normalized start acceleration, constants[3] = normalized end velocity, constants[4] = normalized end acceleration)
PROF_Poly5	21	General polynomial 5th order with arbitrary polynomial coefficients (constants[1 - 6] = polynomial coefficients a0 - a5)

Data type MC_SEGMOD

Name	Value	Description
Standard	0	No end of cycle signalling
REVLOCK_Neg	1	End of cycle signal after leaving segment in negative direction
REVLOCK_Pos	2	End of cycle signal after leaving segment in positive direction
REVLOCK_Both	3	End of cycle signal after leaving segment in any direction

5.2.6   MC_JoinSegments

This function joins two valid segments in the CAM table. Segments are set regarding the boundary conditions of the next and the previous segments. This function can not be called in MSD Servo Drive state "operation enabled". Changed segments will be initialized at switching to state "operation enabled".

For the selectable profile modes see **MCB_SetProfileData**.
For the selectable segment modes see **MCB_SetProfileData**.

NewIndex NextIndex / PrevIndex	New index of the segment (UINT) Following / previous index of this segment; NextIndex and PrevIndex need to be defined valid previously (UINT)
MasterDistance	Segment: Distance of the master (UDINT, increments)
SlaveDistance	Segment: Distance of the slave (DINT, increments)
ProfileMode	Segment: Profile mode; data type **MC_PROFILES** (INT)
SegmentMode	Segment mode of this segment (INT, data type: MC_SEGMOD)
Lambda	Inflexion point parameter (REAL; set lambda to 0.5 for a symmetrical profile)
Return value	Error-ID (INT) regarding list in the appendix

5.2.7 MC_ModifySegment

This function is used to modify online an existing segment. The modify segment must be initialised first and valid. This function will only be executed in MSD Servo Drive state "operation enabled".
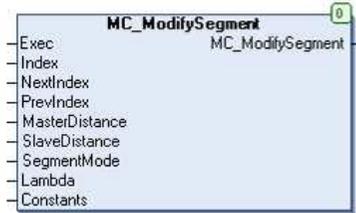
Index	Index of an existing segment (UINT)
NextIndex / PrevIndex	Following / previous index of this segment; NextIndex and PrevIndex need to be defined valid previously (UINT)
MasterDistance	Distance of the master of this segment (UDINT)
SlaveDistance	Distance of the slave of this segment (DINT)
SegmentMode	Segment: Segment mode; Data type MC_SEGMOD (INT)
Lambda	Inflexion point parameter (REAL, set lambda to 0.5 for a symmetrical profile)
Constants	Possibly needed constants to define the segment, depending on the profile mode (ARRAY [1..10] of REAL)
Return value	Error-ID (INT) regarding list in the appendix

5.2.8 MC_SegmentValid

This function checks if a segment is valid.

SegmentIndex	Index of the index to check (UINT)
Return value	"true", if the segment is valid (BOOL)

5.2.9 MC_GetSegment

Returns the actual segment of the ECAM.
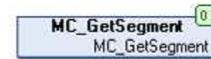
5.2.10 MC_GetSegmentData

This function returns the data of a specified segment.

Index	Index of the segment to read the data (UINT)
NextIndex / PrevIndex	Following / previous index of the segment (UINT)
MasterDistance	Distance of the master of the segment (UDINT)
SlaveDistance	Distance of the slave of the segment (DINT)
ProfileMode	Segment: Profile mode; data type **MC_PROFILES** (INT)
SegmentMode	Mode of the segment, data type **MC_SEGMOD** (INT)

5.2.11   MC_GetSegmentSlaveValues

This function gives back information about a slave segment with a relative position to a specific master position in the specific segment.

If the chosen master position is out of the segment master distance, the corresponding slave position can not be calculated. In this case the function returns an error.

Index   Index of the calculation (UINT)
MasterPos   Master position for calculation (UDINT)
SlavePos   Calculated slave position (DINT)
Return value   Error-ID (INT) regarding list in the appendix

5.3   Special segment creation

The segment creation functions are calculating the polynomial coefficients regarding boundary conditions. These conditions are predefined, except the fact that the position on the start point of the segment is zero in any case.

All user defined boundary conditions are related relative to the master values. The result of the boundary condition function can be used as an input for the following function.

Following functions are available:

Poly2: polynomial 2nd order with quadratic position, linear velocity, a continuous acceleration. Two boundary conditions needs to be defined by the user.

Poly3: polynomial 3rd order with cubic position, quadratic velocity, linear acceleration and continuous jerk. Three boundary conditions need to be defined by the user.

NewIndex   New index of the segment (UINT)
NextIndex   Following index of this segment (UINT)
PrevIndex   Previous index of this segment (UINT)
MasterAmplitude   Amplitude of the master (UDINT; value in increments)
MasterDistance   Distance of the master of this segment (UDINT, increments)
SlaveDistance   Distance of the slave of this segment (DINT, increments)
SegmentMode   Segment mode; data type: **MC_SEGMOD** (INT)
Lambda   Inflexion point parameter (REAL, set value to 0.5 for symmetrical profile)
Vel0, Vel1 Acc,   (Resulting) relative start / end velocity in RPM (REAL)
Acc0, Acc1   (Resulting) relative constant / start / end acceleration in RPM/s (REAL)
Jerk   (Resulting) relative constant jerk in RPM/s^2 (REAL)
Return value   Error-ID (INT) regarding list in the appendix

Note: The boundary condition code (last 4 / 6 characters in the function name) is grouped in combinations of two characters, describing the type (see below).

MC_CreateSegment_Poly2_DSAC

- 2nd Order Polynominal
- (D)istance of the (S)lave of this segment
- (C)onstant (A)cceleration defined

Creates a 2nd order polynomial segment with defined slave distance and constant acceleration

MC_CreateSegment_Poly2_DSV0

- 2nd Order Polynominal
- (D)istance of the (S)lave of this segment
- (V0) Relative Start velocity defined

Creates a 2nd order polynomial segment, defined slave distance and start velocity

MC_CreateSegment_Poly2_V0AC

- 2nd Order Polynominal
- (V0) Relative Start velocity defined
- (C)onstant (A)cceleration defined

Creates a 2nd order polynomial segment with defined start velocity and constant acceleration

MC_CreateSegment_Poly2_V0V1

- 2nd Order Polynominal
- (V0) Relative Start velocity defined
- (V1) Relative End velocity defined

Creates a 2nd order polynomial segment with defined start and end velocity

MC_CreateSegment_Poly3_A0A1V0

- 3rd Order Polynominal
- (A0) Relative start Acceleration defined
- (A0) Relative end Acceleration defined
- (V0) Relative Start velocity defined

Creates a 3rd order polynomial segment with defined start velocity, start acceleration and end acceleration

MC_CreateSegment_Poly3_A0A1V0

- 3rd Order Polynominal
- (A0) Relative start Acceleration defined
- (A0) Relative end Acceleration defined
- (V0) Relative Start velocity defined

Creates a 3rd order polynomial segment with defined slave distance, start acceleration and end acceleration

MC_CreateSegment_Poly3_DSV0A0

- 3rd Order Polynominal
- (D)istance of the (S)lave of this segment
- (V0) Relative Start velocity defined
- (A0) Relative start Acceleration defined

Creates a 3rd order polynomial segment with defined slave distance, start velocity and start acceleration

MC_CreateSegment_Poly3_DSV0V1

- 3rd Order Polynominal
- (D)istance of (S)lave of this segment
- (V0) Relative Start velocity defined
- (V1) Relative End velocity defined

Creates a 3rd order polynomial segment with defined slave distance, start velocity and end velocity

MC_CreateSegment_Poly3_V0A0A1

- 3rd Order Polynominal
- (V0) Relative Start velocity defined
- (A0) Relative start Acceleration defined
- (A1) Relative end Acceleration defined

Creates a 3rd order polynomial segment with defined start velocity, start acceleration and end acceleration

MC_CreateSegment_Poly3_V0A0JC

- 3rd Order Polynominal
- (V0) Relative Start velocity defined
- (A0) Relative start Acceleration defined
- Relative (C)onstant (J)erk

Creates a 3rd order polynomial segment with definable start velocity, start acceleration and constant jerk

6   MSD PLC Sample Programs

Sample programs are available, showing how to use the internal PLC in velocity and speed control mode.

The purpose of these sample programs is to show basic functions of the MSD PLC.

Sample programs can be ordered at Moog support: drives-support@moog.com.
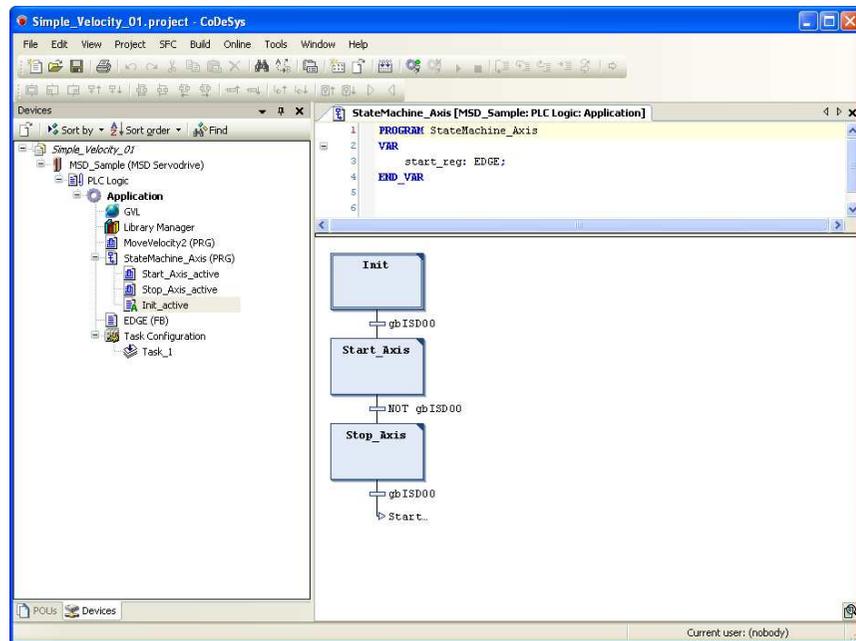
Appendix

A1   Abbreviations

Data type	Description
CAM	A mechanical linkage which translates motion. In the meanwhile CAM functionality is also realized in electronics / software
CFC	Continuous Flow Chart - graphical programming language used for PLCs
CoDeSys	Controller Development System - a development environment for programming controller applications according to the international industrial standard IEC 61131-3
CPU	Central Processing Unit
ECAM	Electronic CAM - electronic version of a linkage which translates motion
EGEAR	Electronic Gear - electronic solution for a gear ratio between a master and a slave
GPOC	Gain, Phase and Offset Correction - system for effective suppression of dominant systematic errors
I/O	Input / Output Interface
IEC 61131	IEC 61131 is an IEC standard for Programmable logic controllers (PLCs). It was known as IEC 1131 before the change in numbering system by IEC.
MDA	Moog Drive Administrator - software for the parameterization of the MSD Servo Drive
MSC II	Moog Servo Controller - a PLC developed by Moog
MSD	Moog Servo Drive
MSD AC-AC	Moog Servo Drive with AC input
MSD DC-AC	Moog Servo Drive with DC input
NVRAM	Non volatile memory
PCON	Position Controller
PDO	Process Data Object - protocol frame for cyclic transfer in CAN / EtherCAT field bus systems
PLC	Programmable Logic Controller - a digital computer used for automation of electromechanical processes

A2   Datatypes

Following data types are used *by the functions* above.

Data type	Description
(BOOL)	Boolean value (true / false)
(INT)	integer value (16 bit)
(UINT)	unsigned integer value (16 bit)
(DINT)	double integer value (32 bit)
(UDINT)	unsigned double integer value (32 bit)
(REAL)	floating point value

A3   MSD PLC Error Codes

(Value) Error name	Description
(0) ERRORID_NoError	No error
(1) ERRORID_ReadProtected	Parameter is read protected
(2) ERRORID_WriteProtected	Parameter is write protected
(3) ERRORID_InUse	Parameter is currently in use
(4) ERRORID_IndexError	Invalid field index
(5) ERRORID_InvalidValue	Value range violation
(6) ERRORID_StringSizeError	String is too long
(7) ERRORID_UnknownDatatype	Data type is unknown
(8) ERRORID_StructDefError	Invalid parameter definition
(9) ERRORID_InvalidDatatype	Invalid data type found
(10) ERRORID_SizeError	Undersized destination data memory
(11) ERRORID_NoParaById	No parameter with this ID found
(12) ERRORID_SystemError	Unknown system error
(13) ERRORID_CorruptBackupVal	Backup value is corrupt
(14) ERRORID_ParaServerLabel	Parameter server exception
(15) ERRORID_ParaServerMsg	Data transfer via parameter server failed
(16) ERRORID_ConversionMdt2Iec	Conversion of parameter to chosen data type not possible
(17) ERRORID_NoDirectParaAccess	Direct parameter access only via process data
(18) ERRORID_TaskInfo	Parameter access via invalid task

(100) ERRORID_RangeViolation	• Input value out of range • The chosen master offset is greater than the master distance of the selected segment • The synchronous position / braking position is bigger than the amplitude of the local master • Not implemented coupling mode selected • No valid axis reference selected • No valid reverse lock mode selected • No valid master speed filter selected • The denominator is zero • The normalized numerator or denominator of the reduced ratio is greater than 65536 • Create segment function: the slave distance / master distance is set to zero and the chosen profile mode is not PROF_Stop or PROF_Standstill
(101) ERRORID_IllegalFBCall	• An instance of a function block has been called multiple times in a single PLC cycle • The function is called in an wrong drive mode • ECAM via MSD PLC is not active • The mode of synchronized motion is not set to ECAM/EGEAR via MSD
(102) ERRORID_FBWrongTask	Instances of function blocks must not be called in more than one task
(103) ERRORID_FBBufferOverFlow	Function block buffer overflow
(104) ERRORID_CmdNotAllowed	• Function / command is not allowed in actual MSD Servo Drive state • Virtual master start / stop: the virtual master is always active / inactive • Segment modification: the segment is not initialised or always active • Probe function is used but not configured • ECAM is not asynchronous

(200) ERRORID_MC_POWER_ NotReadyToSwitchOn (201)	Drive is not in state "ready to switch on" (mandatory according to CiA402 profile) Control location is not assigned to PLC
ERRORID_MC_POWER_ CtrlLocNotPLC	The function MC_Power is already called
(202) ERRORID_MC_POWER_ AnotherPLCActive (300) ERRORID_AxisRefNot Allowed	AxisRef is not allowed here; for a single axis implementation only AXIS_REF_LocalSlave is allowed for Slave, and AXIS_REF_LocalMaster is allowed for master (exception: see MC_MasterConfig)
(301) ERRORID_CAMTableError	• The selected segment is not valid • MC_JoinSegments: the segment which is defined by NextIndex or PrevIndex is not valid • The chosen index or any following index in the predefined cycle is not valid • PROF_Stop: a selected / reached segment in the cycle is a stop segment • CAMIn/CAMTableSelect: CAMTable not valid (segment not available)
(302) ERRORID_CAMTableLocked	• Current CAM Table is locked and may not be changed without preceding CAMOut • MC_ModifySegment: the modification of this segment is always active • ChangeMeth: change of the CAM table is locked and the state of the ECAM is not asynchronous
(303) ERRORID_SegmentInitFailed	The creation of a segment is not possible with the chosen boundary conditions
(305) ERRORID_MasterError	Master could not be started: • The master is not valid or the master is not defined yet • Virtual master is selected but not valid initialised; MCB_VirtualMasterSetPara needs to be used for initialisation.

TAKE A CLOSER LOOK.

Moog solutions are only a click away. Visit our worldwide Web site for more information and the Moog facility nearest you.

Moog GmbH
Hanns-Klemm-Straße 28
D-71034 Böblingen
Telefon +49 7031 622 0

www.moog.com/industrial
drives-support@moog.com

Technical alterations reserved.

The contents of our documentation have been compiled with greatest care and in compliance with our present status of information.

Nevertheless we would like to point that this document cannot always be updated parallel to the technical further development of our products.

Information and specifications may be changed at any time. For information on the latest version please refer to drives-support@moog.com.

ID no: CB15237-001, Rev. 1.4, 10/2024