User's manual
Edition 12/2019
Release H

Automation of the AKD® Drive With a Siemens S7-1200/1500 PLC and Modbus connection

Programmed with TIA Portal V13

Largo Brughetti 1/B2 • 20813 Bovisio Masciago (MB) • Italia • Tel: +39 0362 594260 • www.kollmorgen.com • P.IVA 02965490960 • Cod. Fisc. 02491750234

Record of Document Revisions

Revision	*Remarks*
A 11/2014	First version
B 12/2014	Reworked version, Modbus data access direct in function bloc instead of public Merker area
C 02/2015	reworked edition, AKD_MBComm changes to Software V0.8
D 02/2015	Scaling corrected New function: read motion task
E 09/2015	Corrected the current scaling Converted to TIA V13 SP1
F 03/2016	Rework of the watchdog functionality Adding S7-1200 with Firmware 4.1
G 05/2016	Drive warnings added to the interface (AKD Firmware M_01-14-05-000)
H 12/2019	Update to the latest feedback from the filed

Notes
For better readability of the code all variables have a of lower case letter prefix, the definition is as following.

Prefix	Size	Description
i	Unknown	Input
o	Unknown	Output
b	1 Bit	Bool
by	8 Bit	Byte
w	16 Bit	Word
dw	32 Bit	Double Word
	Unknown	Variable struct

Contents

Introduction

The covered system contains an AKD-P drive and an S7-1200 PLC.
All Automation functions are controlled by the PLC. The functions covered are the following:

- Software Enable/Disable
- Emergency stop
- Reset of drive faults
- Jog
- Homing (with direct positioning or with the AKD home functions)
- Starting of a pre-defined motion task
- Setup motion tasks
- Read motion tasks
- Change operation mode
- Direct read or write access to parameters

Each Function is controlled by specific bits and variables and the actual values of the drive are read back.

This manual should help to understand the relation between parameter and with a few simple steps allow to do some motion.

	**Never use the TIA-Project „AKD_MBComm_Modules.zap13, AKD_MBCommS1500_Modules.zap13“ unchanged in an application. The application is intended as an example how an AKD can be integrated into a TIA-Project. This project example always has to be changed according to the real application.**
	**KOLLMORGEN srl cannot be held liable for any damage caused by the use of the TIA-Project „AKD_MBComm_Modules.zap13, AKD_MBCommS1500_Modules.zap13“ or any parts of it.**

Configuration AKD

The connection between AKD-P (plug X11) and PLC (plug X1P1) is done by Modbus TCP hereafter called Modbus. The basic setup of the AKD has to be done in Workbench. As the Modbus communication uses port 502 and Workbench port 23 both connections can be on at the same time.

In the PLC the instruction MB_Client is used to exchange data between the two devices. To speed up reading in the AKD-P all data to be read are dynamically mapped at the end of the Modbus register.

In the Terminal the following macro has to be executed „MBCommSet.macro“. This is used to setup the basic settings for the Modbus communication. The corresponding commands can also be typed in by hand.

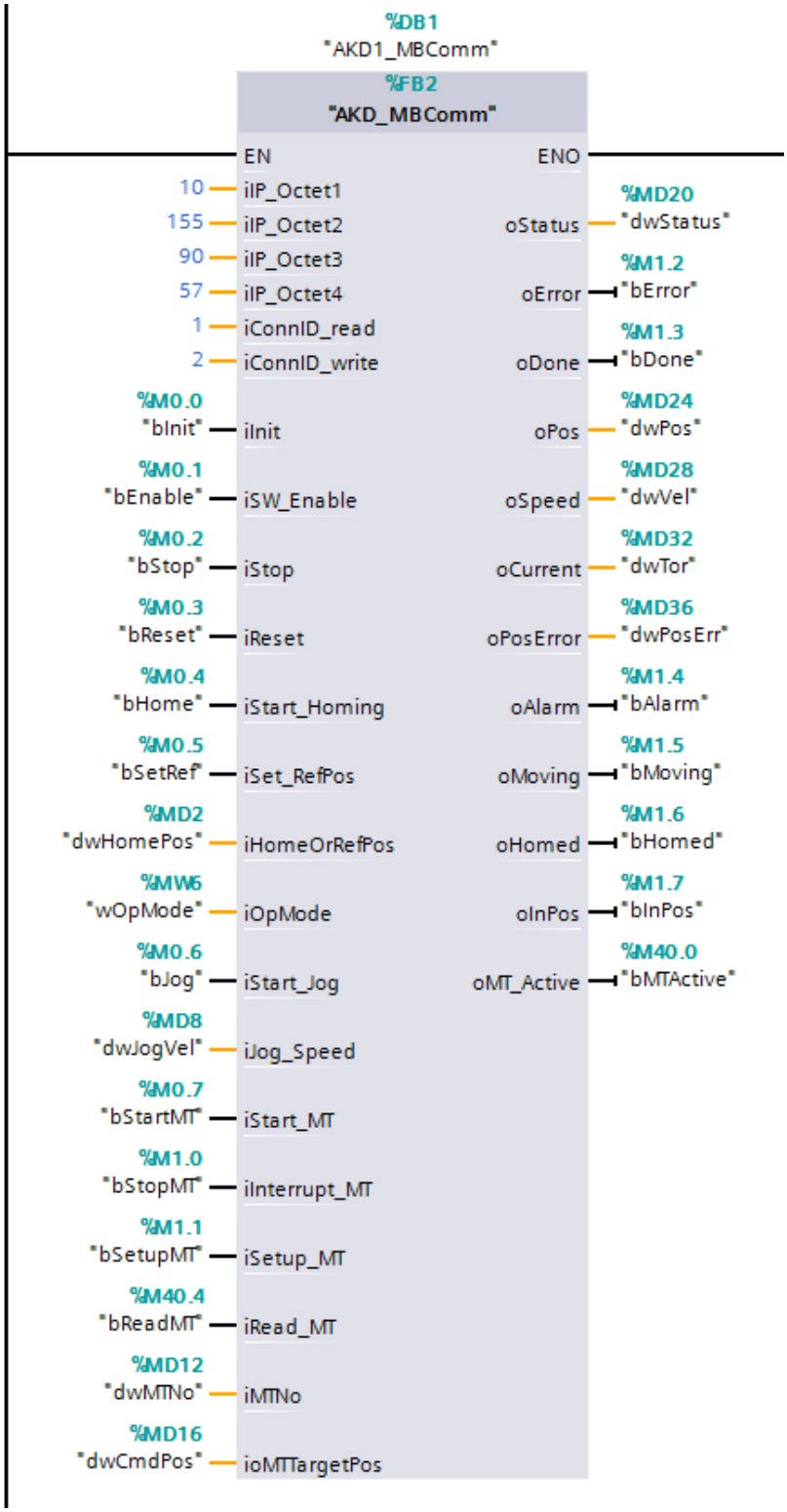
List of the Parameter which have to be set for the AKD communication. This can be done by the „MBCommSet.macro“.

Terminal Command	Description
MODBUS.DYNMAP 1	Enable dynamic mapping
MODBUS.CLRDYNMAP	Clear the dynamic mapping area
MODBUS.ADDR8192 945	Write Parameter 945 to the Modbus address 8192: MODBUS.DRVSTAT
MODBUS.ADDR8193 268	DRV.MOTIONSTAT High Byte
MODBUS.ADDR8194 269	DRV.MOTIONSTAT Low Byte
MODBUS.ADDR8195 2072	PL.FB_32 High Byte
MODBUS.ADDR8196 2073	PL.FB_32 Low Byte
MODBUS.ADDR8197 2066	PL.ERR_32 High Byte
MODBUS.ADDR8198 2067	PL.ERR_32 Low Byte
MODBUS.ADDR8199 954	MODBUS.FAULT1 High Byte
MODBUS.ADDR8200 955	MODBUS.FAULT1 Low Byte
MODBUS.ADDR8201 856	VL.FB High Byte
MODBUS.ADDR8202 857	VL.FB Low Byte
MODBUS.ADDR8203 432	IL.FB High Byte
MODBUS.ADDR8204 433	IL.FB Low Byte
MODBUS.ADDR8205 1582	DRV.WARNING1 High Byte
MODBUS.ADDR8206 1583	DRV.WARNING1 Low Byte
MODBUS.ADDR8300 2060	MT.P_32 High Byte
MODBUS.ADDR8301 2061	MT.P_32 Low Byte
MODBUS.ADDR8302 566	MT.V High Byte
MODBUS.ADDR8303 567	MT.V Low Byte
MODBUS.ADDR8304 2056	MT.ACC_32 High Byte
MODBUS.ADDR8305 2057	MT.ACC_32 Low Byte
MODBUS.ADDR8306 2058	MT.DEC_32 High Byte
MODBUS.ADDR8307 2059	MT.DEC_32 Low Byte
MODBUS.ADDR8308 532	MT.CNTL High Byte
MODBUS.ADDR8309 533	MT.CNTL Low Byte
MODBUS.ADDR8310 546	MT.NEXT High Byte
MODBUS.ADDR8311 547	MT.NEXT Low Byte
MODBUS.DYNMAP 0	Disable dynamic mapping
MODBUS.PSCALE 20	Scale positon resolution PL.FB_32
SM.MODE 0	Service motion mode 0
SM.T1 0	Service motion time T1 0ms
SM.T2 0	Service motion time T2 0ms
DIN.LCMD1 DRV.STOP;DRV.CMDDELAY 500;DRV.DIS	Command buffer 1: Stop, delay of 500ms, disable the drive
DIN3.MODE 9	Switch digital input 3 to command buffer
DIN3.PARAM 1	Command buffer 1
DRV.NVSAVE	Save the changes to the drive.

AKD - PLC Connection

Configuration PLC

The Function bloc **AKD_MBComm** is used as the interface to the user in the PLC. This bloc controls the read and write, which is executed with the MB_Client function bloc. The data bloc MBR to read and MBW to write is used.



Variable name	Typ	Direction	Description
EN	Bool	In	Call of the DB
iIP_OctetX	USInt	In	IP-Address of the connected AKD
iConnID_read	UInt	In	Connection ID of the read command **(has to be unique)**
iConnID_write	UInt	In	Connection ID of the write command **(has to be unique)**
iInit	Bool	In	Disconnection of all Modbus communication
iSW_Enable	Bool	In	Switches the software enable/disable
iStop	Bool	In	Stop of all motion execution
iReset	Bool	In	Reset of drive faults
iStart_Homing	Bool	In	Starts the homing procedure as setup in the AKD
iSet_RefPos	Bool	In	Sets the actual position as referenced
iHomeOrRefPos	DWord	In	Position which is set after homing or set reference
iOpMode	Int	In	Operation mode of the AKD
iStart_Jog	Bool	In	Starts the jogging
iJog_Speed	DWord	In	Jog Speed
iStart_MT	Bool	In	Starts the corresponding motion task in the drive
iInterrupt_MT	Bool	In	Interrupts an active motion task
iSetup_Mt	Bool	In	Setup a motion task in the AKD
iRead_MT	Bool	In	Read all motion task parameter from the AKD
iMTNo	DWord	In	Sets the number of the motion task for setup or start
ioMTTargetPos	DWord	In/Out	Target position of a motion task setup
ENO	Bool	Out	DB call done
oStatus	DWord	Out	Actual status of the AKD
oError	Bool	Out	Connection or function bloc faults
oDone	Bool	Out	Modbus write command execution successful
oPos	DWord	Out	Actual position
oSpeed	DWord	Out	Actual speed
oCurrent	DWord	Out	Actual current
oPosError	DWord	Out	Actual position error
oAlarm	Bool	Out	Actual AKD Error
oMoving	Bool	Out	Axis is moving
oHomed	Bool	Out	Axis is homed
oInPos	Bool	Out	Axis is within position window of the motion task
oMT_Active	Bool	Out	Motion task active
wMBR_Data[0..15]	Word	Static	Array of the read Modbus data
dwMBW_Data[0..5]	DWord	In/Out	Array of the write Modbus data
bMBR_Req	Bool	Static	Starts a read request
dwMBR_Addr	DWorrd	Static	Read address of the Modbus data
bMBR_Busy	Bool	Static	Indicates an active read
bMBR_Done	Bool	Static	Indicates a successful read
bMBR_Error	Bool	Static	Indicates read connection errors
wMBR_Status	Word	Static	Indicates the status of the read connection
bMBW_Req	Bool	Static	Starts a write request
dwMBW_Addr	DWord	Static	Write address of the Modbus data
bMBW_Busy	Bool	Static	Indicates an active write
bMBW_Done	Bool	Static	Indicates a successful write
bMBW_Error	Bool	Static	Indicates write connection errors
wMBW_Status	Word	Static	Indicates the status of the write connection
dwMBW_ChechForWrite	DWord	Static	Checks for the next order to be written
wMBW_DataLenght	Word	Static	Length of the write data register
bDriveNeedsDisable	Bool	Static	Indicates the Software enable of the AKD
bJogIsActive	Bool	Static	Indicates the start of a jog move
bHomingIsActive	Bool	Static	Indicates the start of a homing move
bSetRefPosIsActive	Bool	Static	Indicates the start of a direct reference
bStartMTIsActive	Bool	Static	Indicates the start of a motion task
bOldResetState	Bool	Static	Comparison to input Reset
wOldOpMode	Word	Static	Comparison to input OpMode

dwOldJogSpeed	DWord	Static	Comparison to input JogSpeed
dwOldHomeOrRefPos	DWord	Static	Comparison to input RefPos
dwSetupMT	DWord	Static	Local status of the setup for a new motion task
bMBWInProgress	Bool	Static	Write is in progress
MBR	MB_Client	Static	Modbus communication bloc to read
MBW	MB_Client	Static	Modbus communication bloc to write
bSetup_MTDone	Bool	Static	Setup of motion task in the AKD done
bMTInterrupted	Bool	Static	An active motion task has been stopped and waits for the continue
bResetInPos	Bool	Static	Resets the InPos for minimum one cycle with every start or continue of a motion task
bMBR_Init	Bool	Static	Disconnect and reset the read connection
bMBW_Init	Bool	Static	Disconnect and reset the write connection
TimeStamp	DTL	Static	System time
dwMillSec	DInt	Static	Mill seconds of the system time
dwMBRTimeStamp	DInt	Static	Time stamp for the call of the read function
wLocalWDTime	Word	Static	Timeout limit for the read function
bWDError	Bool	Static	Watch dog timeout in the read connection
dwTempRunTime	DInt	Static	Runtime of the latest read call
bReadMTDone	Bool	Static	Read motion task completed
dwReadMT	DWord	Static	Local status of the read motion task
bWDActive	Bool	Static	Local status of the communication supervision watchdog
bTempRetVal	Int	Static	RD_SYS_T return value (0 = ok)
loldTimeStamp	DTL	Static	Local copy of Timestamp to supervise rising values

In wMBR_Data Word Array all the data read with Modbus are stored. The write data are stored in the dwMBW_Data double word array. Where dwMBW_Data[0] is always used and is set within the function bloc. dwMBW_Data[1] – dwMBW_Data[5] are only used to setup a motion task and the corresponding value has to be set directly in the data field. The following list shows the definition of the individual variables within the array.

Status Data

Variable	*Direction*	*Size*	*Description*	*Bit*	*Description*	*Bit*	*Description*
wMBR_Data[0]	IN	Word	MODBUS.DRVSTAT	3	Negative HW limit	0	Drive active
				4	Positive SW limit	1	STO status
				5	Negative SW limit	2	Positive HW limit
wMBR_Data[1]	IN	Word	DRV.MOTIONSTAT	0	E-Stop error in the drive		
				1	Service motion active		
				2	Invalid Motion task		
				3	Motion task in position		
wMBR_Data[2]	IN	Word	DRV.MOTIONSTAT	4	Home error	0	Motion task active
				5	Slave gear synch	1	Home found
				6	Gear mode active	2	Home finished
				7	E-Stop in progress	3	Home active
wMBR_Data[3]	IN	Word	PL.FB_32 high byte				
wMBR_Data[4]	IN	Word	PL.FB_32 low byte				
wMBR_Data[5]	IN	Word	PL.ERR_32 high byte				
wMBR_Data[6]	IN	Word	PL.ERR_32 low byte				
wMBR_Data[7]	IN	Word	MODBUS.FAULT1 high byte				
wMBR_Data[8]	IN	Word	MODBUS.FAULT1 low byte				

wMBR_Data[9]	IN	Word	VL.FB high byte
wMBR_Data[10]	IN	Word	VL.FB low byte
wMBR_Data[11]	IN	Word	IL.FB high byte
wMBR_Data[12]	IN	Word	IL.FB low byte
wMBR_Data[13]	IN	Word	DRV.WARNING1 high byte
wMBR_Data[14]	IN	Word	DRV.WARNING1 low byte

Control Data

Variable	***Direction***	***Size***	***Description***	***Access***
dwMBW_Data[0]	OUT	DWord	Data for the next write command	Internal
dwMBW_Data[1]	OUT	DWord	MT.V, motion task speed	External
dwMBW_Data[2]	OUT	DWord	MT.ACC, motion task acceleration	External
dwMBW_Data[3]	OUT	DWord	MT.DEC, motion task deceleration	External
dwMBW_Data[4]	OUT	DWord	MT.CNTL, motion task control word	External
dwMBW_Data[5]	OUT	DWord	MT.NEXT, next motion task	External

Notes

ConnID_read and ConnID_write are not allowed to have the same value. If there are more than one axis the numbers for all ConnID must all be different!

The data bloc inputs are treated by priority, if there is more than one signal at the time. The highest priority is one.

If the **communication fails** the AKD **doesn't notice** anything. With a digital input setup as 9-command buffer, the communication supervision can be done with one hardwired signal. On the negative edge a DRV.STOP/DRV.DIS with or without delay can be programmed to handle this situation. An example of this is in the MBCommSet.macro done with the digital input three and command buffer one.
The PLC supervises the connection with a 500ms watchdog. The *oError* bit of the function bloc indicates a trigger. If the fault stays the *iInit* bit can be used to force a connection reestablishment. To do so the *iInit* has to be high for one second.

A setup motion task in the AKD can be read with the *iRead_MT*. As this function uses the MBR the actual values of the function bloc are not updated for one cycle during execution of the *iRead_MT*. The read values are stored in *iMTTargetPos* and the dwMBW_Data[1..5] array, according to the *iMTTragetPos* function.

If TIA-Portal is online the communication to the drive is disturbed, which leads to inconsistent delays (9ms – 80ms). The tested configuration had a delay between to read cycle of 9ms without TIA-Portal. If a second AKD is added the time goes up to 18ms and 26ms for an application with three AKD. These are best possible data, as no other PLC code is executed at the same time.

The used AKD-P00306-NAEC is running on Firmware M_01-18-00-004.

This project has been done with Siemens CPU1214 DC/DC/DC and can need some changes due to other configurations.

Detailed input description

iIP_OctetX
Sets the IP-Address of the connected AKD drive.

iConnID_read/write
Sets the connection number for the Modbus client. Each connection in the same PLC needs to have a unique number. The ConnID_read and the ConnID_write must be different!

iInit
If active the Modbus communication is disconnected. If an Error (communication problems) occurs the *iInit* bit can help reset the connection. The *iInit* Bit has to be high till the *oError* goes low, at least for a minimum of 300ms. If the *iInit* is set low with the *oError* Low immediatly, the MB_Client might not be stopped completely. In this situation the MB_Client will fail with reestablish connection and produce some random status changes, which can cause the Modbus port on the AKD to fail.

iSW_ENABLE
0 = disables the AKD, priority 1.
1= enables the AKD, priority 5.

iReset
Reset of all active drive fault. The instruction gets executed once per rising edge, priority 2.

iStop
Stop of all motion in progress without disable the AKD, priority 3.

iStart_Homing
Starts the homing, according to the setup done in the Workbench. The instruction gets executed once per rising edge, priority 7.

iSet_RefPos
Sets the actual position as reference the homing is complete with this. The AKD doesn't need to be enabled for this execution. The instruction gets executed once per rising edge, priority 9.

iHomeOrRefPos
The positon which is set after a homing or set reference, this value is used by the *iStart_homing* and the *iSet_RefPos*. The correct scaling of the value can be seen in the section "value scaling", priority 6.

iOpMode
Operation mode switching of the AKD, priority 4.
0 = Torque mode
1 = Velocity mode
2 = Position mode

iStart_Jog
Starts a move on the fixed speed *iJog_Speed*. In *iOpMode* = 2 the axis needs to be stopped to change the speed. The jog uses the service motion function of the AKD. In *iOpMode* = 1 a speed change on the fly is possible. The command uses the VL.CMDU functionality of the AKD, priority 12.

iJog_Speed
Sets the speed for the jog move. The correct scaling of the value can be seen in the section “value scaling”, priority 11.

iStart_MT
Starts a motion task which has be pre-configured in the AKD on positon *iMTNo*. The instruction gets executed once per rising edge, priority 14.

iInterrupt_MT
Stops an active motion task. On the edge to zero the motion task is going to be continued. If a motion task is stopped by *iStop*, the AKD Software enable gets disable, or a *iReset* is executed the continuing option is suspended, see the interrupt sequence graphic, priority 15.

iSetup_MT
Gives the possibility to setup a complete motion task from the PLC. The motion task is saved in the AKD on the number *iMTNo*, the number must not be zero! The setup can take up to 300ms. It’s possible to setup a motion task during execution and start it directly after finishing of the old task. The values which are used for setup the motion task into the AKD, at the designated motion task number, are explained with *iMTTargetPos*. The instruction gets executed once per rising edge, priority 13.

iRead_MT
Reads all the data of the motion task *iMTNo* and stores the values according to *iMTTragetPos* in the dwMBW_Data Array, priority 17.

iMTNo
Gives the number of the motion task to be executed with *iSetup_MT* or *iRead_MT*. To do a setup the number has to be unequal zero.

iMTTargetPos
A Motion Task contains target position, speed, acceleration, deceleration, the motion task control word and the following motion task number. All this parameter excluded the target position have to be setup directly at the corresponding pointer address. The correct scaling of the value can be seen in the section “value scaling”.
dwMBW_Data[1] = Speed (MD1034), MT.V
dwMBW_Data[2] = Acceleration (MD1038), MT.ACC
dwMBW_Data[3] = Deceleration (MD1042), MT.DEC
dwMBW_Data[4] = Control word (MD1046), MT.CNTL 0 = absolute, 1= relative
dwMBW_Data[5] = following motion task, this is only active if MT.CNTL Bit 4 is TRUE

For a more detail instruction manual use the AKD workbench help at the designated commands.

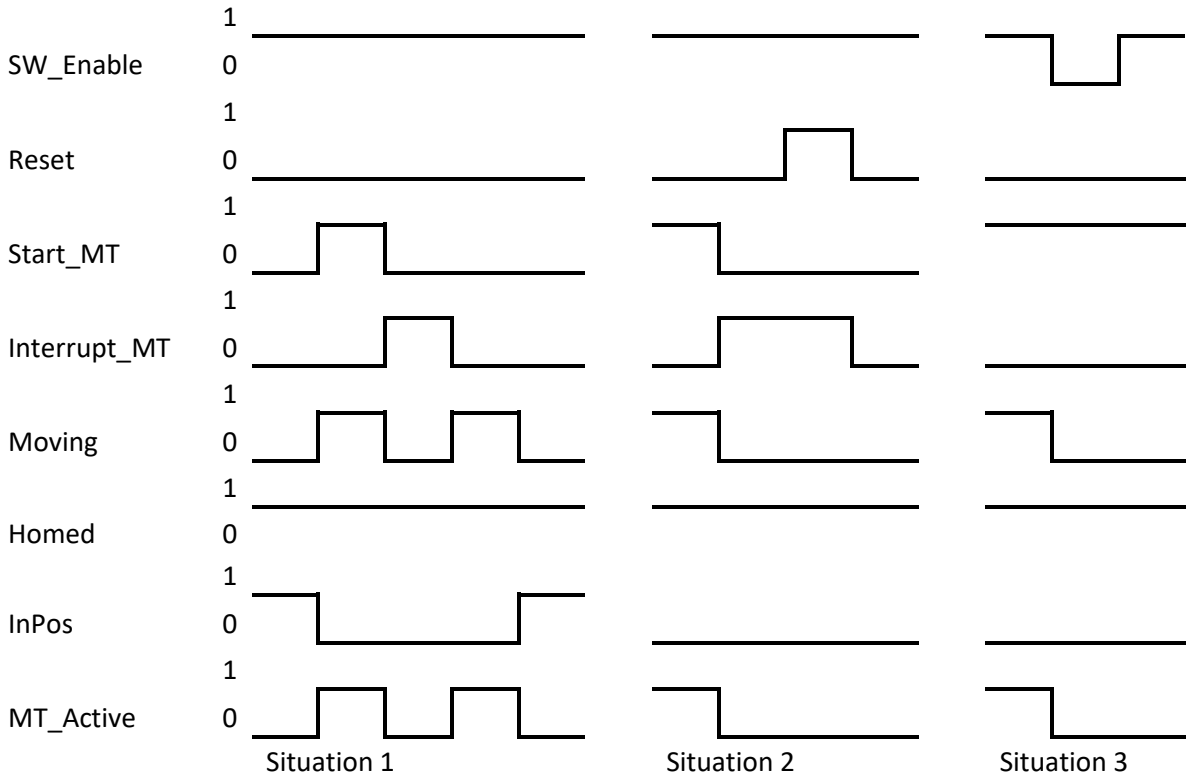
Value scaling

Scaling for position values
The Position represents a Modbus scaled per revolution.
Position in the AKD x MODBUS.PSCALE / (MODBUS.PIN / MODBUS.POUT) = value in the PLC
The calculation changes a little bit if the UNIT.PROTRA = 3 is set in the AKD.
Position in the AKD x MODBUS.PSCALE / (MODBUS.PIN / MODBUS.POUT) ^2 = value in the PLC

Scaling for velocity values
The velocity represents a Modbus scaled revolution per second.
Velocity in the AKD x MODBUS.PSCALE / 60 = value in the PLC

Scaling for acceleration values
The acceleration represents a Modbus scaled revolution per minute per second.
Acceleration in the AKD x MODBUS.PSCALE / 60 = value in the PLC

Interrupt sequence



1: An *iInerrupt_MT* stops an active motion task, the motion task will be continued when the Interrupt_MT goes to zero again. If the falling edge of *iInerrupt_MT* and *iStart_MT* are at the same time the new motion task will be executed, continue will be suspended in this case.

2: An *iInterrupt_MT* stops an active motion Task. The motion task will not be continued if a *iReset* or *iStop* is executed during Interrupt.

3: A *iStop* or *iSW_Enable* stops a motion task. The motion task will not be continued. The same happens if the AKD has an error state or the STO is switched off.

Detailed output description

oStatus
Indicates the actual status of the AKD.

Bit 0	Drive active
Bit 1	STO Status
Bit 2	Positive hardware limit reached
Bit 3	Negative hardware limit reached
Bit 4	Positive software limit reached
Bit 5	Negative software limit reached
Bit 6	Invalid motion task activated
Bit 8 - 19	AKD fault code (identical with the Display)
Bit 20 - 31	AKD warning code (identical with the Display)

oError
Indicates an error in the communication or the function bloc. The read and the write connection faults are indicated here. In addition the read connection has a watchdog timeout of 500ms. This timeout is connected to the busy of the MBR. The error bit is reset automatically if the fault is solved, or if the *iInit* bit disconnects the Modbus client.

oDone
Indicates the successful transmission of a command. The *oDone* bit is zero during execution of the Modbus write command. If the MBR is done successful the done bit goes one.

oPosition
Actual axis position as 32 Bit DWord, PL.FB_32. The correct scaling of the value can be seen in the section “value scaling”.

oSpeed
Actual axis speed as 32 Bit DWord, VL.FB. The correct scaling of the value can be seen in the section “value scaling”.

oCurrent
Actual current value as 32 Bit DWord, IL.FB in mA. With a maximum resolution of AKD peak current / 20130 [Arms].

oPositionError
Actual axis position error as 32 Bit DWord, PL.ERR_32. The correct scaling of the value can be seen in the section “value scaling”.

oAlarm
Indicates errors in the drive. Only AKD faults are shown.

oMoving
Indicates an active jog or motion task.

oHomed
Indicates that the homing or set reference positon has been done successful. This must be TRUE to start any motion task.

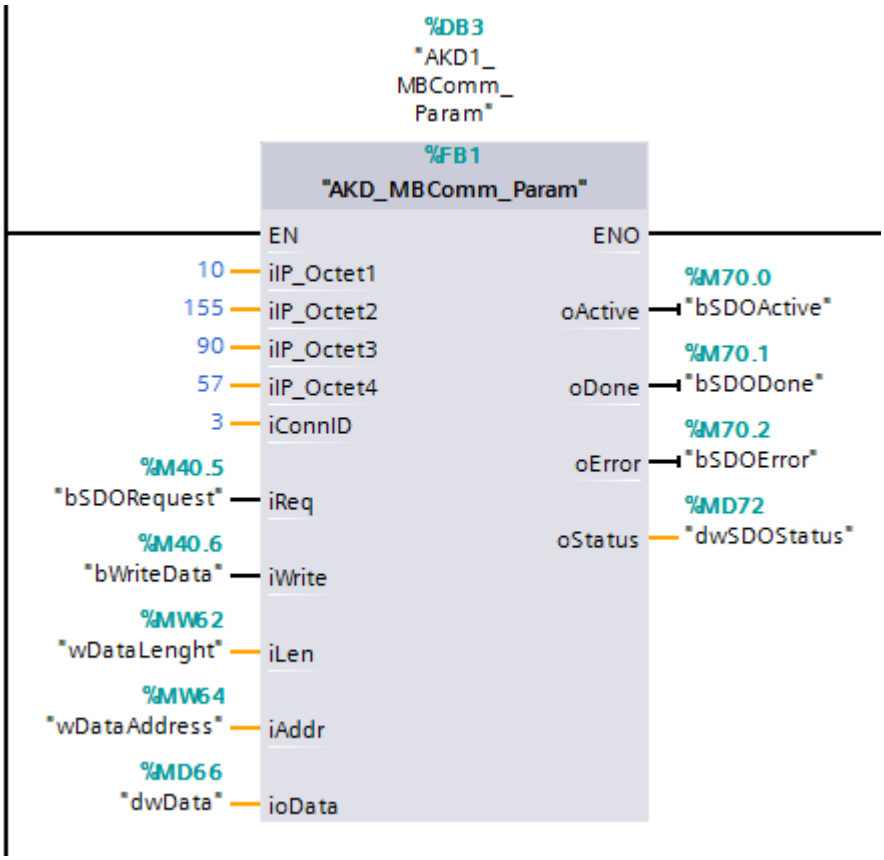
oInPos
Indicates that the actual position is within MT.POSWND, of the target position, of the executing motion task. The movement doesn't have to be finished at this moment. If a motion task gets started or continued the *oInPos* always goes zero for a minimum of one cycle. This also happens if the motion task called in the AKD gives an error or doesn't do any movement at all.

oMT_Active
Indicates the execution of a motion task.

Direct parameter access

Configuration PLC

The direct parameter access works with the second function bloc **AKD_MBComm_Param**. With this a direct access to all available Modbus parameter can be done. It's possible to access multiple AKD after each other with the same data bloc. The only thing that needs to change is the IP-address of the corresponding AKD.



Variable name	Typ	Direction	Description
EN	Bool	In	Call of the DB
iIP_OctetX	USInt	In	IP-Address of the connected AKD
iConnID	UInt	In	Connection ID **(unique)**
iReq	Bool	In	Start of a read or write command
iWrite	Bool	In	Defines if the parameter has to be read or written 0 = Read, 1 = Write
iLen	Integer	In	Defines the length of the register
iAddr	Integer	In	Defines the Modbus address of the parameter
ioData	DWord	In/Out	Write = 1, Date to be written to the parameter Write = 0, Data read form the parameter
ENO	Bool	Out	DB call done
oActive	Bool	Out	MBRW is in progress
oDone	Bool	Out	MBRW is done successfully
oError	Bool	Out	MBRW has an error or some wrong input parameter are set
oStatus	DWord	Out	Status of the MBRW

dwDataArray[0..3]	DWord	Static (In/Out)	Data array for the modbus communication
MBRW	MBClient	Static	MB_Client function to read or write
bMBRW_Req	Bool	Static	Internal call of the MBRW
bMBRW_Discn	Bool	Static	Disconnect the AKD
wMBRW_Mode	USInt	Static	Internal setting of the read or write mode
dwMBRW_Address	UDInt	Static	Internal address of the parameter
wMBRW_Len	UInt	Static	Internal length of the register
bMBRW_Done	Bool	Static	Internal status MBRW is done
bMBRW_Busy	Bool	Static	Internal status MBRW is in progress
bMBRW_Error	Bool	Static	Internal status MBRW has an error
wMBRW_Status	Word	Static	Internal status of the MBRW
bDoNextRW	Bool	Static	Command in Progress
bValidFault	Bool	Static	Wrong input parameters
wOldIP_OctetX	USInt	Static	Internal IP-Address of the connected AKD

Notes

A connection is directly done with the enable of the data bloc.

If there is an error the connection is gone be terminated and only to be reconnected after the error has been cleared. Possible causes of an error can be wrong input parameters, error in the MB_Client function or the change of the IP-address.

A fault input is: if the length of the register is ≤ 0 or if the Modbus address is < 0 or > 9999.

ioData always represents the dwDataArray[0]. In read mode (*iWrite* = FALSE) *ioData* cannot be changed from outside the function bloc.

If a register of more than two words needs to be executed the access for the data has to be done directly with the *dwDataArray[]*. This double word array is configured for four DWord but can be extended to any need. The maximum length is 123 Word, which means 61 DWord. In write mode the *dwDataArray[0]* will always be overwritten by the *ioData* input. Therefor the first DWord has to be set via the Interface while the following can be directly access via the data bloc.

Detail in- and output description

iReq
Starts a read or write command. The instruction gets executed once per rising edge.

iWrite
Command to set the read or write.
0 = Read
1 = Write

iLen
Length of the register. At the moment a maximum of two register (one DWord) is accessible with the interface. For longer registers the data area has to be changed accordingly.
2 = one 32-Bit or less Variable
4 = one 64-Bit Variable

iAddr
Address of the Modbus register, according to the AKD manual. For Parameter with a total length of 64-Bit, there are special 32-Bit versions (Mapping of 64-Bit-Parametern to 32-Bit-Parameter).

ioData
Output of the read data, the value is shown till a new request is started. During a request the data is zero.
Input if the data are written. In write mode the data are not changed at all in the function block. If the *iReq* is used to activate a command in the AKD with no value, the *ioData* value has to be unequal to zero.

oActive
Indicates a request in progress.

oDone
Is set one with every successful request. The Data are valid form this moment. Done is reset with a new request or a termination of the connection.

oError
Indicates an MBRW error or a validation fault. A validation fault is caused by wrong register addresses, invalid register length or change of IP-address. If an error occurs the connection is terminated.

oStatus
Direct image of the Modbus connection status (MBRW). Detailed help is found in the TIA-Portal help under MB_Client.

Changes for a S1500 or S1200 PLC with Firmware ≥ 4.1

The project AKD_MBComm_Modules.zap13 is used to connect with a S1200 PLC. With the project AKD_MBCommS1500_Modules.zap13 a connection to a S1500 PLC can be done. The functionality and interface stays the same. The changes done are explain here after shortly and should help to understand the small differences between the two versions.

The MB_Client bloc has been changed with S1200 V4.0 and S1500. Now the internal signals are working better and the connection is setup with a separate structure. Therefor the variables MBR_Connect, MBW_Connect and MBRW_Connect of the data type TCON_IP_v4 are added.

With these changes, it's now possible to change the IP-address and Connection ID to an active connection.

With the S1500 the delay between two reads is not anymore depending too much on the axis count. With one axis the delay time measured has been 8ms, for two axis 10ms and for three axis 12ms. These are best possible data, as no other PLC code is executed at the same time.

This project has been done with a Siemens CPU1513-1 PN and can need some changes due to other configurations.

About KOLLMORGEN
Kollmorgen is a leading provider of motion systems and components for machine builders. Through world-class knowledge in motion, industry-leading quality and deep expertise in linking and integrating standard and custom products, Kollmorgen delivers breakthrough solutions that are unmatched in performance, reliability and ease-of-use, giving machine builders an irrefutable marketplace advantage.

For assistance with your application needs, visit www.kollmorgen.com or contact us at:

KOLLMORGEN srl
Largo Brughetti 1/B2
20813 Bovisio Masciago, Italia
Web www.kollmorgen.com
Email mil-info@kollmorgen.com
Tel.: +39 - 0362 - 594260
Fax: +39 - 0362 - 594263