

Totally Integrated Automation Portal								
AKD_PN_TG400_v03 [FB1]								
AKD_PN_TG400_v03 Properties								
General								
Name	AKD_PN_TG400_v03	Number	1	Type	FB	Language	SCL	
Numbering	automatic							
Information								
Title	AKD_PN_TG400_TIA13_v03	Author	jcoleman	Comment	This function block supports Profinet communication with the AKD drive using Telegram 400. It is written in TIA V13. For more information, please see the AKD Profinet Manual and the supporting documentation for this function block.	Family	TG400	
Version	3.0	User-defined ID						
Name	Data type	Offset	Default value	Accessible from HMI	Visible in HMI	Setpoint	Comment	
▼ Input								
iID	HW_IO	0.0	16#0	True	True	False	Hardware identifier	
iInit	Bool	2.0	false	True	True	False	initialize config for TG400	
iEStop	Bool	2.1	false	True	True	False	immediate quick stop and disable	
iSW_Enable	Bool	2.2	false	True	True	False	enable/disable	
iStop	Bool	2.3	false	True	True	False	stop all movement	
iReset	Bool	2.4	false	True	True	False	reset drive faults	
iStart_Homing	Bool	2.5	false	True	True	False	start homing	
iStart_Jog	Bool	2.6	false	True	True	False	start jog	
iStart_Move	Bool	2.7	false	True	True	False	start motion task	
iMT_Pause	Bool	3.0	false	True	True	False	pause active motion task	
iMT_MoveType	Word	4.0	16#0	True	True	False	motion task control word 0 = relative; 1 = absolute	
iMTNum	Word	6.0	16#0	True	True	False	motion task number 0x8000 = direct motion task	
iCmdPos	DWord	8.0	16#0	True	True	False	commanded position for motion task or homing	
iCmdVel	DWord	12.0	16#0	True	True	False	commanded velocity for jog or direct motion task	
iCmdAcc	Word	16.0	16#0	True	True	False	commanded acceleration for jog or direct motion task	
iCmdDec	Word	18.0	16#0	True	True	False	commanded deceleration for jog or direct motion task	
▼ Output								
oInitDone	Bool	20.0	false	True	True	False	initialization status - 0 while configuring TG400, 1= done	
oStatus	DWord	22.0	16#0	True	True	False	ZSW1	
oPos	DWord	26.0	16#0	True	True	False	actual position	
oVel	Word	30.0	16#0	True	True	False	actual velocity x * 12000 rpm / 2^15	
oCurrent	Word	32.0	16#0	True	True	False	actual current x *DRV.IPEAK / 2^14	
oEnabled	Bool	34.0	false	True	True	False	Drive power on	
oError	Bool	34.1	false	True	True	False	fault in function block or communication	
oMoving	Bool	34.2	false	True	True	False	axis enabled and moving	
oHomed	Bool	34.3	false	True	True	False	axis homed	
oInPos	Bool	34.4	false	True	True	False	axis within in-position window	
oFault	Bool	34.5	false	True	True	False	drive fault	
oWarning	Bool	34.6	false	True	True	False	drive warning	
InOut								
▼ Static								
▼ TG_400	"UDT_AKD_TG400"	36.0		True	True	False		
▼ Send	Struct	0.0		False	False	False		
STW1	Int	0.0	0	False	False	False	Control word 1	
SATZANW	Int	2.0	0	False	False	False	Motion task selection	
STW2	Int	4.0	0	False	False	False	Control word 2	
MDI_TARPOS	Int	6.0	0	False	False	False	MDI motion task target position	
MDI_TARPOS_1	Int	8.0	0	False	False	False	MDI motion task target position	
MDI_VELOCITY	Int	10.0	0	False	False	False	MDI motion task velocity	
MDI_VELOCITY_1	Int	12.0	0	False	False	False	MDI motion task velocity	
MDI_ACC	Int	14.0	0	False	False	False	MDI motion task acceleration	
MDI_DEC	Int	16.0	0	False	False	False	MDI motion task deceleration	
MDI_MOD	Int	18.0	0	False	False	False	MDI motion task control word	
HOME_DIST	Int	20.0	0	False	False	False	Home distance	
HOME_DIST_1	Int	22.0	0	False	False	False	Home distance	
IDT_DATAITEM_I_12	Int	24.0	0	False	False	False	Process data word input 13	
IDT_DATAITEM_I_13	Int	26.0	0	False	False	False	Process data word input 14	
IDT_DATAITEM_I_14	Int	28.0	0	False	False	False	Process data word input 15	
IDT_DATAITEM_I_15	Int	30.0	0	False	False	False	Process data word input 16	
▼ Receive	Struct	32.0		False	False	False		

Totally Integrated Automation Portal		
<pre>0103 #bSDOReq := TRUE; 0104 #bSDOWriteDone := FALSE; 0105 #wSDOStatus := 3; //proceed to Step 3 0106 END_IF; 0107 // setup the array data and start WRREC 0108 IF #wSDOStatus = 3 THEN //Step 3 0109 #bySDODataArray[0] := 16#fe; //Request Reference 0110 #bySDODataArray[1] := 16#02; //2 for write PNU 0111 #bySDODataArray[2] := 16#00; //Axis# is 0 for AKD 0112 #bySDODataArray[3] := 16#02; // Number of parameters to write at a time 0113 // Set PNU 915 subindex 4 0114 #bySDODataArray[4] := 16#10; //Attribute is 0x10 0115 #bySDODataArray[5] := 16#01; //Number of elements =1 0116 #bySDODataArray[6] := 16#03; //0x394 for PNU 916 0117 #bySDODataArray[7] := 16#94; //PNU 916 0118 #bySDODataArray[8] := 16#00; //subindex 0119 #bySDODataArray[9] := 16#04; //subindex 4 for 4th word of telegram 0120 // Set PNU 915 subindex 5 0121 #bySDODataArray[10] := 16#10; //Attribute is 0x10 0122 #bySDODataArray[11] := 16#01; //Number of elements =1 0123 #bySDODataArray[12] := 16#03; //0x394 for PNU 916 0124 #bySDODataArray[13] := 16#94; //PNU 916 0125 #bySDODataArray[14] := 16#00; //subindex 0126 #bySDODataArray[15] := 16#05; //subindex 5 for 5th word of telegram 0127 // To PNU 6 (VL.FB) 0128 #bySDODataArray[16] := 16#42; //Format =0x42 for Word 0129 #bySDODataArray[17] := 16#01; //Number of values 0130 #bySDODataArray[18] := 16#00; //data 0131 #bySDODataArray[19] := 16#06; //data - Signal #6 0132 // To PNU 52 (IL.FB) 0133 #bySDODataArray[22] := 16#42; //Format =0x42 for Word 0134 #bySDODataArray[23] := 16#01; //Number of values 0135 #bySDODataArray[24] := 16#00; //data 0136 #bySDODataArray[25] := 16#34; //data - Signal #52 0137 0138 #SDOWrite(REQ := #bSDOReq, //do the PNU write 0139 ID := #iID, 0140 INDEX := 47, //AKD supports "Record Data 47" 0141 LEN := 50, //data length 0142 BUSY => #bSDOWriteBusy, 0143 DONE => #bSDOWriteDone, 0144 ERROR => #bSDOWriteError, 0145 RECORD := #bySDODataArray); 0146 0147 IF #bSDOWriteBusy THEN 0148 #bSDOReq := FALSE; 0149 ELSIF #bSDOWriteDone THEN 0150 #wSDOStatus := 4; //proceed to Step 4 0151 END_IF; 0152 END_IF; 0153 0154 // Set OpMode 0155 // Clear the Array and start the request 0156 IF #wSDOStatus = 4 THEN //Step 4 0157 FILL_BLK(IN := 0, 0158 COUNT := 64, 0159 OUT => #bySDODataArray[0]); 0160 #bSDOReq := TRUE; 0161 #bSDOWriteDone := FALSE; 0162 #wSDOStatus := 5; 0163 END_IF; 0164 // Set OpMode = 2 Position 0165 IF #wSDOStatus = 5 THEN 0166 #bySDODataArray[0] := 16#fd; //Request Reference 0167 #bySDODataArray[1] := 16#02; //2 for write PNU 0168 #bySDODataArray[2] := 16#00; //Axis# is 0 for AKD 0169 #bySDODataArray[3] := 16#01; // Number of parameters to write at a time 0170 // Set PNU 930 subindex 0 (DRV.OPMODE) 0171 #bySDODataArray[4] := 16#10; //Attribute is 0x10 0172 #bySDODataArray[5] := 16#01; //Number of elements =1 0173 #bySDODataArray[6] := 16#03; //0x3A2 for PNU 930 - opmode 0174 #bySDODataArray[7] := 16#A2; //PNU 930 0175 #bySDODataArray[8] := 16#00; //subindex 0 0176 #bySDODataArray[9] := 16#00; //subindex 0 0177 // To 2 (position mode) 0178 #bySDODataArray[10] := 16#42; //Format =0x42 for Word 0179 #bySDODataArray[11] := 16#01; //Number of values 0180 #bySDODataArray[12] := 16#00; //data 0181 #bySDODataArray[13] := 16#02; //data - vlaue = 2 for position mode 0182 0183 #SDOWrite(REQ := #bSDOReq, //do the PNU write 0184 ID := #iID, 0185 INDEX := 47, //AKD supports "Record Data 47" 0186 LEN := 14, //data length 0187 BUSY => #bSDOWriteBusy, 0188 DONE => #bSDOWriteDone, 0189 ERROR => #bSDOWriteError, 0190 RECORD := #bySDODataArray);</pre>		

Totally Integrated Automation Portal		
<pre>0191 0192 IF #bSDOWriteBusy THEN 0193 #bSDOReq := FALSE; 0194 ELSIF #bSDOWriteDone THEN 0195 #wSDOStatus := 16; // setup complete - skip to process 0196 #oInitDone := 1; //telegram configuration complete 0197 END_IF; 0198 END_IF; 0199 END_IF; 0200 0201 // Read ZSW and set output values 0202 IF NOT #iInit AND #wSDOStatus > 15 THEN 0203 #wPNReadStatus := DPRD_DAT(LADDR := #iID, 0204 RECORD => #TG_400.Receive); 0205 IF #wPNReadStatus = 0 THEN 0206 #bPNReadError := FALSE; 0207 0208 #oStatus.%W0 := #TG_400.Receive.ZSW1; 0209 IF #wSDOStatus > 16 THEN 0210 #oStatus.%W1 := #wSDOReadValue; 0211 ELSE 0212 #oStatus.%W1 := 0; 0213 END_IF; 0214 #oPos.%W0 := #TG_400.Receive.XIST_A_1; 0215 #oPos.%W1 := #TG_400.Receive.XIST_A; 0216 #oVel := #TG_400.Receive.NIST_A; 0217 #oCurrent := #TG_400.Receive.ITIST_GLATT; 0218 0219 #oEnabled := #TG_400.Receive.ZSW1.%X0 AND #TG_400.Receive.ZSW1.%X1 AND #TG_400.Receive.ZSW1.%X2 AND NOT #TG_400.Receive.ZSW1.%X6; 0220 #oFault := #TG_400.Receive.ZSW1.%X3; 0221 #oMoving := NOT #TG_400.Receive.ZSW1.%X13 OR #TG_400.Receive.ZSW1.%X14 OR #TG_400.Receive.ZSW1.%X12; 0222 #oHomed := #TG_400.Receive.ZSW1.%X11; 0223 IF #TG_400.Send.STW1.%X6 THEN // Reset the InPos Flag for a minimum of one write cycle 0224 #oInPos := FALSE; 0225 ELSE 0226 #oInPos := #TG_400.Receive.ZSW1.%X10; 0227 END_IF; 0228 #oWarning := #TG_400.Receive.ZSW1.%X7; 0229 ELSE 0230 #bPNReadError := TRUE; 0231 END_IF; 0232 0233 END_IF; 0234 0235 // Do the write always as the zero Telegramm needs to be send on a init aswell 0236 #wPNWriteStatus := DPWR_DAT(LADDR := #iID, 0237 RECORD := #TG_400.Send); 0238 0239 // set STW1 and values for the AKD 0240 IF NOT #iInit AND #wSDOStatus > 15 THEN 0241 IF #wPNWriteStatus = 0 THEN 0242 #bPNWriteError := FALSE; 0243 0244 #TG_400.Send.STW1.%X10 := TRUE; 0245 IF #TG_400.Receive.ZSW1.%X9 THEN // PLC has control 0246 0247 // Do a EStop 0248 IF #iEStop THEN 0249 #TG_400.Send.STW1.%X2 := FALSE; 0250 #bEstop := TRUE; 0251 END_IF; 0252 0253 // Do a reset 0254 IF #iReset AND NOT #bResetDone THEN 0255 #wSDOStatus := 16; 0256 #TG_400.Send.STW1.%X7 := TRUE; 0257 #bResetDone := TRUE; 0258 #bEstop := FALSE; 0259 #wStartReset := #TG_400.Receive.ZSW2; 0260 END_IF; 0261 IF ABS(#wStartReset - #TG_400.Receive.ZSW2) > 2 THEN 0262 #TG_400.Send.STW1.%X7 := FALSE; 0263 END_IF; 0264 0265 // Enable the drive 0266 IF #iSW_Enable AND NOT #bEstop AND NOT (#bSWEnabled AND #TG_400.Receive.ZSW1.%X6 OR #TG_400.Re- ceive.ZSW1.%X3) THEN 0267 IF #TG_400.Receive.ZSW1.%X6 THEN // Switch on inhibited 0268 #TG_400.Send.STW1.%X1 := TRUE; 0269 #TG_400.Send.STW1.%X2 := TRUE; 0270 ELSE 0271 IF #TG_400.Receive.ZSW1.%X0 THEN // Ready for switiching on 0272 #TG_400.Send.STW1.%X0 := TRUE; 0273 IF #TG_400.Receive.ZSW1.%X1 THEN // Switched on 0274 #TG_400.Send.STW1.%X3 := TRUE; 0275 IF #TG_400.Receive.ZSW1.%X2 THEN // Operation 0276 #TG_400.Send.STW1.%X12 := TRUE; //turn on real time jogging mode</pre>		

Totally Integrated Automation Portal		
0277	#bSWEnabled := TRUE;	
0278		
0279	// Do a Stop	
0280	IF #iStop THEN	
0281	#TG_400.Send.STW1.%X4 := FALSE; // Stop active motion task	
0282	#TG_400.Send.STW1.%X5 := FALSE; // Stop active motion task	
0283	#TG_400.Send.STW1.%X8 := FALSE; // Stop jog 1	
0284	#TG_400.Send.STW1.%X9 := FALSE; // Stop jog 2	
0285	#TG_400.Send.STW1.%X11 := FALSE; // Stop homing	
0286	//#TG_400.Send.STW1.%X12 := FALSE; // Stop real time jogging mode	
0287	ELSE	
0288		
0289	// Do home	
0290	IF #iStart_Homing AND NOT #bHomeStarted THEN	
0291	#TG_400.Send.HOME_DIST := #iCmdPos.%W1;	
0292	#TG_400.Send.HOME_DIST_1 := #iCmdPos.%W0;	
0293	#TG_400.Send.STW1.%X11 := TRUE;	
0294	#wHomingStartPoint := #TG_400.Receive.ZSW2;	
0295	#bHomeStarted := TRUE;	
0296	ELSE	
0297	IF #TG_400.Receive.ZSW1.%X11 AND #TG_400.Receive.ZSW1.%X13 AND (ABS(#wHoming-	
0298	StartPoint - #TG_400.Receive.ZSW2) > 5) THEN	
0299	#TG_400.Send.STW1.%X11 := FALSE;	
0300	END_IF;	
0301		
0302	// Jogging	
0303	IF #iStart_Jog AND #iCmdVel <> 0 THEN	
0304	#TG_400.Send.STW1.%X8 := TRUE;	
0305	//#TG_400.Send.STW1.%X12 := TRUE;	
0306		
0307	// if the speed input is negative set Bit 13 for reverse direction	
0308	IF #iCmdVel.%X31 = 1 THEN	
0309	#TG_400.Send.STW1.%X13 := TRUE;	
0310	#TG_400.Send.MDI_VELOCITY := (16#ffff - #iCmdVel.%W1 + 1);	
0311	#TG_400.Send.MDI_VELOCITY_1 := (16#ffff - #iCmdVel.%W0 + 1);	
0312	ELSE	
0313	#TG_400.Send.STW1.%X13 := FALSE;	
0314	#TG_400.Send.MDI_VELOCITY := #iCmdVel.%W1;	
0315	#TG_400.Send.MDI_VELOCITY_1 := #iCmdVel.%W0;	
0316	END_IF;	
0317		
0318	#TG_400.Send.MDI_ACC := #iCmdAcc;	
0319	#TG_400.Send.MDI_DEC := #iCmdDec;	
0320	ELSE	
0321	#TG_400.Send.STW1.%X8 := FALSE;	
0322	//#TG_400.Send.STW1.%X12 := FALSE;	
0323		
0324	// Interrupt a move	
0325	IF #iMT_Pause THEN	
0326	#TG_400.Send.STW1.%X5 := FALSE; // intermediate stop, motion is	
0327	paused and will continue when the pause is lifted	
0328	ELSE	
0329	#TG_400.Send.STW1.%X5 := TRUE; // No intermediate stop	
0330	END_IF;	
0331		
0332	// Reset the rising edge of the start bit to be read for the next task	
0333	IF #TG_400.Receive.ZSW1.%X12 OR ABS(#wStartMove - #TG_400.Receive.ZSW2) >	
0334	62 THEN //if start command has been acknowledged, or >62 telegram transmissions	
0335	#bStartMoveDone := TRUE; // the process to start the move has been	
0336	completed	
0337		
0338	#TG_400.Send.STW1.%X6 := FALSE; //turn off the start bit	
0339	END_IF;	
0340	// Move	
0341	IF #iStart_Move AND NOT #bStartMove THEN	
0342	#TG_400.Send.MDI_MOD := #iMT_MoveType; // 1 = absolute, 0 = relative	
0343	#TG_400.Send.SATZANW := #iMTNum; // 0x8000 = direct motion task,	
0344	otherwise select a motion task from the AKD	
0345		
0346	#TG_400.Send.MDI_TARPOS := #iCmdPos.%W1;	
0347	#TG_400.Send.MDI_TARPOS_1 := #iCmdPos.%W0;	
0348	#TG_400.Send.MDI_VELOCITY := #iCmdVel.%W1;	
0349	#TG_400.Send.MDI_VELOCITY_1 := #iCmdVel.%W0;	
0350	#TG_400.Send.MDI_ACC := #iCmdAcc;	
0351	#TG_400.Send.MDI_DEC := #iCmdDec;	
0352	#TG_400.Send.STW1.%X4 := TRUE; // Do not reject traversing task	
0353	#TG_400.Send.STW1.%X5 := TRUE; // No intermediate stop	
0354	#TG_400.Send.STW1.%X6 := TRUE; // Start the move	
0355		
0356	#bStartMove := TRUE; // the process to start the move has been started	
0357	but not completed	
0358		
0359	#bStartMoveDone := FALSE; // the process to start the move has not	
0360	been completed	
0361		
0362	#wStartMove := #TG_400.Receive.ZSW2;	
0363	END_IF; //If Start move	
0364		
0365	END_IF; //If Start Jog	
0366	END_IF; //If Start Homing	
0367	END_IF; // If Stop	
0368	END_IF; // If ZSW1 bit2 Operation Enabled	

Totally Integrated Automation Portal		
0358	END_IF; // If ZSW1 bit1 Switched On	
0359	END_IF; // If ZSW1 bit0 Ready for Switch On	
0360	END_IF; // If ZSW1 bit6 Switch-On Inhibited	
0361		
0362	// If the axis gets disabled all the corresponding motion signals have to be reset	
0363	ELSE	
0364	#bSWEnabled := FALSE;	
0365		
0366	#TG_400.Send.STW1.%X0 := FALSE;	
0367	#TG_400.Send.STW1.%X1 := FALSE;	
0368	#TG_400.Send.STW1.%X2 := FALSE;	
0369	#TG_400.Send.STW1.%X3 := FALSE;	
0370	#TG_400.Send.STW1.%X4 := FALSE;	
0371	#TG_400.Send.STW1.%X5 := FALSE;	
0372	#TG_400.Send.STW1.%X6 := FALSE;	
0373	#TG_400.Send.STW1.%X8 := FALSE;	
0374	#TG_400.Send.STW1.%X9 := FALSE;	
0375	#TG_400.Send.STW1.%X11 := FALSE;	
0376	#TG_400.Send.STW1.%X12 := FALSE;	
0377	END_IF; //If Enable	
0378	END_IF; // ZSW1 bit9 PLC has control	
0379	ELSE	
0380	#bPNWriteError := TRUE;	
0381	END_IF; //If #wPNWriteStatus = 0	
0382		
0383	// Read drive faults and warning on occurens	
0384	IF #oFault THEN	
0385	// If alarm is not read yet: start read	
0386	IF NOT #wSDOStatus.%X6 AND NOT #bSDOStatusReadActive THEN	
0387	#wSDOStatus := 64;	
0388	#wSDOAddress := 16#09AD; // Read PNU 2477 (DRV.FAULT1)	
0389	#bSDOStatusReadActive := TRUE;	
0390	END_IF;	
0391		
0392	// If warning is not read yet: start read	
0393	ELSIF #oWarning THEN	
0394	IF NOT #wSDOStatus.%X5 AND NOT #bSDOStatusReadActive THEN	
0395	#wSDOStatus := 32;	
0396	#wSDOAddress := 16#0AE7; // Read PNU2791 (DRV.WARNING1)	
0397	#bSDOStatusReadActive := TRUE;	
0398	END_IF;	
0399	ELSE	
0400	IF NOT #bSDOStatusReadActive THEN	
0401	#wSDOStatus := 16;	
0402	END_IF;	
0403	END_IF; //IF #oFault	
0404		
0405	// start the reading signal	
0406	IF #wSDOStatus >= 32 THEN	
0407	// Always clean out old values first	
0408	IF #bSDOWriteBusy THEN	
0409	#bSDOReq := FALSE;	
0410	ELSE	
0411	FILL_BLK(IN := 0,	
0412	COUNT := 64,	
0413	OUT => #bySDODataArray[0]);	
0414	#bSDOReq := TRUE;	
0415	#bSDOWriteDone := FALSE;	
0416	END_IF;	
0417	// Start the read cylce to the corrspoding PNU	
0418	IF #wSDOStatus = 32 OR #wSDOStatus = 64 THEN	
0419	#bySDODataArray[0] := #wSDOStatus.%B0;	
0420	#bySDODataArray[1] := 16#01;	
0421	#bySDODataArray[2] := 16#00;	
0422	#bySDODataArray[3] := 16#01;	
0423		
0424	#bySDODataArray[4] := 16#10;	
0425	#bySDODataArray[5] := 16#01;	
0426	#bySDODataArray[6] := #wSDOAddress.%B1;	
0427	#bySDODataArray[7] := #wSDOAddress.%B0;	
0428	#bySDODataArray[8] := 16#00;	
0429	#bySDODataArray[9] := 16#00;	
0430		
0431	#SDOWrite(REQ := #bSDOReq,	
0432	ID := #iID,	
0433	INDEX := 47,	
0434	LEN := 14,	
0435	BUSY => #bSDOWriteBusy,	
0436	DONE => #bSDOWriteDone,	
0437	ERROR => #bSDOWriteError,	
0438	RECORD := #bySDODataArray);	
0439	IF #bSDOWriteDone THEN	
0440	#wSDOStatus := #wSDOStatus + 1;	
0441	END_IF;	
0442	END_IF; //IF #wSDOStatus = 32 OR #wSDOStatus = 64	
0443		
0444	// read the value back	
0445	IF #wSDOStatus = 33 OR #wSDOStatus = 65 THEN	

Totally Integrated Automation Portal

```
0446      #SDORead(REQ := #bSDOReq,
0447                ID := #iID,
0448                INDEX := 47,
0449                MLEN := 14,
0450                BUSY => #bSDOReadBusy,
0451                VALID => #bSDOReadDone,
0452                ERROR => #bSDOReadError,
0453                RECORD := #bySDODataArray);
0454      IF #bSDOReadDone THEN
0455          IF #bySDODataArray[4] = 66 THEN // if the read signal is a Word use Byte 6&7
0456              #wSDOReadValue.%B0 := #bySDODataArray[7];
0457              #wSDOReadValue.%B1 := #bySDODataArray[6];
0458          ELSE // if the read signal is a DWord use Byte 8&9
0459              #wSDOReadValue.%B0 := #bySDODataArray[9];
0460              #wSDOReadValue.%B1 := #bySDODataArray[8];
0461          END_IF;
0462          IF #wSDOReadValue = 0 THEN // If the read value is zero repeat the read cycle
0463              #wSDOStatus := #wSDOStatus - 1;
0464          ELSE
0465              #wSDOStatus := #wSDOStatus + 1;
0466              #bSDOStatusReadActive := FALSE;
0467          END_IF;
0468          END_IF; //IF #bSDOReadDone
0469      END_IF; //IF #wSDOStatus = 33 OR #wSDOStatus = 65
0470  END_IF; //IF #wSDOStatus >= 32
0471 END_IF; //IF NOT #iInit AND #wSDOStatus > 15
0472
0473 // check for rising edge for the input parameters
0474 IF NOT #iReset AND #bResetDone THEN
0475     #bResetDone := FALSE;
0476 END_IF;
0477 IF NOT #iStart_Homing AND #bHomeStarted THEN
0478     #bHomeStarted := FALSE;
0479 END_IF;
0480 IF NOT #iStart_Move AND #bStartMove AND #bStartMoveDone AND NOT #TG_400.Receive.ZSW1.%X12 THEN
0481     #bStartMove := FALSE;
0482     #bStartMoveDone := FALSE;
0483 END_IF;
0484
0485 // fault detection/handling within the bloc
0486 IF #bPNReadError OR #bPNWriteError OR #bSDOReadError OR #bSDOWriteError THEN
0487     #oError := TRUE;
0488     IF #bPNReadError THEN
0489         #oStatus := #oStatus;
0490     END_IF;
0491     IF #bPNWriteError THEN
0492         #oStatus := #oStatus;
0493     END_IF;
0494 ELSIF #wSDOStatus > 15 THEN
0495     #oError := FALSE;
0496 END_IF;
0497
```

Symbol	Address	Type	Comment
#bEstop		Bool	
#bHomeStarted		Bool	
#bPNReadError		Bool	
#bPNWriteError		Bool	
#bResetDone		Bool	
#bSDOReadBusy		Bool	
#bSDOReadDone		Bool	
#bSDOReadError		Bool	
#bSDOReq		Bool	
#bSDOStatusReadActive		Bool	
#bSDOWriteBusy		Bool	
#bSDOWriteDone		Bool	
#bSDOWriteError		Bool	
#bStartMove		Bool	
#bStartMoveDone		Bool	
#bSWEnabled		Bool	
#bySDODataArray		Array	
#iCmdAcc		Word	commanded acceleration for jog or direct motion task
#iCmdDec		Word	commanded deceleration for jog or direct motion task
#iCmdPos.%W0		Word	commanded position for motion task or homing
#iCmdPos.%W1		Word	commanded position for motion task or homing
#iCmdVel		DWord	commanded velocity for jog or direct motion task
#iCmdVel.%W0		Word	commanded velocity for jog or direct motion task
#iCmdVel.%W1		Word	commanded velocity for jog or direct motion task
#iCmdVel.%X31		Bool	commanded velocity for jog or direct motion task
#iEstop		Bool	immediate quick stop and disable
#iID		HW_IO	Hardware identifier
#iInit		Bool	initialize config for TG400
#iMT_MoveType		Word	motion task control word 0 = relative; 1 = absolute
#iMT_Pause		Bool	pause active motion task
#iMTNum		Word	motion task number 0x8000 = direct motion task
#iReset		Bool	reset drive faults
#iStart_Homing		Bool	start homing
#iStart_Jog		Bool	start jog

Totally Integrated Automation Portal		
Symbol	Address	TypeComment
#iStart_Move		Boolstart motion task
#iStop		Boolstop all movement
#iSW_Enable		Boolenable/disable
#oCurrent		Wordactual current x * DRV.IPEAK / 2^14
#oEnabled		BoolDrive power on
#oError		Boolfault in function block or communication
#oFault		Booldrive fault
#oHomed		Boolaxis homed
#olnitDone		Boolinitialization status - 0 while configuring TG400, 1= done
#olnPos		Boolaxis within in-position window
#oMoving		Boolaxis enabled and moving
#oPos.%W0		Wordactual position
#oPos.%W1		Wordactual position
#oStatus		DWordZSW1
#oStatus.%W0		WordZSW1
#oStatus.%W1		WordZSW1
#oVel		Wordactual velocity x * 12000 rpm / 2^15
#oWarning		Booldrive warning
#SDORead		Multi_SFB
#SDOWrite		Multi_SFB
#TG_400.Receive		Struct
#TG_400.Receive.ITIST_GLATT		IntActual current
#TG_400.Receive.NIST_A		IntActual velocity
#TG_400.Receive.XIST_A		IntActual position
#TG_400.Receive.XIST_A_1		IntActual position
#TG_400.Receive.ZSW1		IntStatus word 1
#TG_400.Receive.ZSW1.%X0		BoolStatus word 1
#TG_400.Receive.ZSW1.%X1		BoolStatus word 1
#TG_400.Receive.ZSW1.%X2		BoolStatus word 1
#TG_400.Receive.ZSW1.%X3		BoolStatus word 1
#TG_400.Receive.ZSW1.%X6		BoolStatus word 1
#TG_400.Receive.ZSW1.%X7		BoolStatus word 1
#TG_400.Receive.ZSW1.%X9		BoolStatus word 1
#TG_400.Receive.ZSW1.%X10		BoolStatus word 1
#TG_400.Receive.ZSW1.%X11		BoolStatus word 1
#TG_400.Receive.ZSW1.%X12		BoolStatus word 1
#TG_400.Receive.ZSW1.%X13		BoolStatus word 1
#TG_400.Receive.ZSW1.%X14		BoolStatus word 1
#TG_400.Receive.ZSW2		IntStatus word 2
#TG_400.Send		Struct
#TG_400.Send.HOME_DIST		IntHome distance
#TG_400.Send.HOME_DIST_1		IntHome distance
#TG_400.Send.MDI_ACC		IntMDI motion task acceleration
#TG_400.Send.MDI_DEC		IntMDI motion task deceleration
#TG_400.Send.MDI_MOD		IntMDI motion task control word
#TG_400.Send.MDI_TARPOS		IntMDI motion task target position
#TG_400.Send.MDI_TARPOS_1		IntMDI motion task target position
#TG_400.Send.MDI_VELOCITY		IntMDI motion task velocity
#TG_400.Send.MDI_VELOCITY_1		IntMDI motion task velocity
#TG_400.Send.SATZANW		IntMotion task selection
#TG_400.Send.STW1		IntControl word 1
#TG_400.Send.STW1.%X0		BoolControl word 1
#TG_400.Send.STW1.%X1		BoolControl word 1
#TG_400.Send.STW1.%X2		BoolControl word 1
#TG_400.Send.STW1.%X3		BoolControl word 1
#TG_400.Send.STW1.%X4		BoolControl word 1
#TG_400.Send.STW1.%X5		BoolControl word 1
#TG_400.Send.STW1.%X6		BoolControl word 1
#TG_400.Send.STW1.%X7		BoolControl word 1
#TG_400.Send.STW1.%X8		BoolControl word 1
#TG_400.Send.STW1.%X9		BoolControl word 1
#TG_400.Send.STW1.%X10		BoolControl word 1
#TG_400.Send.STW1.%X11		BoolControl word 1
#TG_400.Send.STW1.%X12		BoolControl word 1
#TG_400.Send.STW1.%X13		BoolControl word 1
#TG_400.Send.STW2		IntControl word 2
#wHomingStartPoint		Word
#wPNReadStatus		Word
#wPNWriteStatus		Word
#wSDOAddress		Word
#wSDOAddress.%B0		Byte
#wSDOAddress.%B1		Byte
#wSDOReadValue		Word
#wSDOReadValue.%B0		Byte
#wSDOReadValue.%B1		Byte
#wSDOStatus		Word
#wSDOStatus.%B0		Byte
#wSDOStatus.%X5		Bool
#wSDOStatus.%X6		Bool
#wStartMove		Int
#wStartReset		Int