

Totally Integrated Automation Portal								
AKD_PN_TG400_v04 [FB1]								
AKD_PN_TG400_v04 Properties								
General								
Name	AKD_PN_TG400_v04	Number	1	Type	FB	Language	SCL	
Numbering	automatic							
Information								
Title	AKD_PN_TG400_TIA13_v04	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	4.0	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								
Name	Data type	Offset	Default value	Accessible from HMI	Visible in HMI	Setpoint	Comment	
ZSW1	Int	0.0	0	False	False	False	Status word 1	
AKTSATZ	Int	2.0	0	False	False	False	Actual motion task selection	
ZSW2	Int	4.0	0	False	False	False	Status word 2	
XIST_A	Int	6.0	0	False	False	False	Actual position	
XIST_A_1	Int	8.0	0	False	False	False	Actual position	
NIST_A	Int	10.0	0	False	False	False	Actual velocity	
ITIST_GLATT	Int	12.0	0	False	False	False	Actual current	
IDT_DATAITEM_O_7	Int	14.0	0	False	False	False	Process data word output 8	
IDT_DATAITEM_O_8	Int	16.0	0	False	False	False	Process data word output 9	
IDT_DATAITEM_O_9	Int	18.0	0	False	False	False	Process data word output 10	
IDT_DATAITEM_O_10	Int	20.0	0	False	False	False	Process data word output 11	
IDT_DATAITEM_O_11	Int	22.0	0	False	False	False	Process data word output 12	
IDT_DATAITEM_O_12	Int	24.0	0	False	False	False	Process data word output 13	
IDT_DATAITEM_O_13	Int	26.0	0	False	False	False	Process data word output 14	
IDT_DATAITEM_O_14	Int	28.0	0	False	False	False	Process data word output 15	
IDT_DATAITEM_O_15	Int	30.0	0	False	False	False	Process data word output 16	
wPNReadStatus	Word	100.0	16#0	True	True	False		
bPNReadError	Bool	102.0	false	True	True	False		
wPNWriteStatus	Word	104.0	16#0	True	True	False		
bPNWriteError	Bool	106.0	false	True	True	False		
bResetDone	Bool	106.1	false	True	True	False		
bHomeStarted	Bool	106.2	false	True	True	False		
bStartMoveDone	Bool	106.3	false	True	True	False		
bEstop	Bool	106.4	false	True	True	False		
wStartReset	Int	108.0	0	True	True	False		
wStartMove	Int	110.0	0	True	True	False		
wSDOStatus	Word	112.0	16#0	True	True	False		
▼ bySDODataArray	Array[0..63] of Byte	114.0		True	True	False		
bySDODataArray[0]	Byte	0.0	16#0	True	True	False		
bySDODataArray[1]	Byte	1.0	16#0	True	True	False		
bySDODataArray[2]	Byte	2.0	16#0	True	True	False		
bySDODataArray[3]	Byte	3.0	16#0	True	True	False		
bySDODataArray[4]	Byte	4.0	16#0	True	True	False		
bySDODataArray[5]	Byte	5.0	16#0	True	True	False		
bySDODataArray[6]	Byte	6.0	16#0	True	True	False		
bySDODataArray[7]	Byte	7.0	16#0	True	True	False		
bySDODataArray[8]	Byte	8.0	16#0	True	True	False		
bySDODataArray[9]	Byte	9.0	16#0	True	True	False		
bySDODataArray[10]	Byte	10.0	16#0	True	True	False		
bySDODataArray[11]	Byte	11.0	16#0	True	True	False		
bySDODataArray[12]	Byte	12.0	16#0	True	True	False		
bySDODataArray[13]	Byte	13.0	16#0	True	True	False		
bySDODataArray[14]	Byte	14.0	16#0	True	True	False		
bySDODataArray[15]	Byte	15.0	16#0	True	True	False		
bySDODataArray[16]	Byte	16.0	16#0	True	True	False		
bySDODataArray[17]	Byte	17.0	16#0	True	True	False		
bySDODataArray[18]	Byte	18.0	16#0	True	True	False		
bySDODataArray[19]	Byte	19.0	16#0	True	True	False		
bySDODataArray[20]	Byte	20.0	16#0	True	True	False		
bySDODataArray[21]	Byte	21.0	16#0	True	True	False		
bySDODataArray[22]	Byte	22.0	16#0	True	True	False		
bySDODataArray[23]	Byte	23.0	16#0	True	True	False		
bySDODataArray[24]	Byte	24.0	16#0	True	True	False		
bySDODataArray[25]	Byte	25.0	16#0	True	True	False		
bySDODataArray[26]	Byte	26.0	16#0	True	True	False		
bySDODataArray[27]	Byte	27.0	16#0	True	True	False		
bySDODataArray[28]	Byte	28.0	16#0	True	True	False		
bySDODataArray[29]	Byte	29.0	16#0	True	True	False		
bySDODataArray[30]	Byte	30.0	16#0	True	True	False		
bySDODataArray[31]	Byte	31.0	16#0	True	True	False		
bySDODataArray[32]	Byte	32.0	16#0	True	True	False		
bySDODataArray[33]	Byte	33.0	16#0	True	True	False		
bySDODataArray[34]	Byte	34.0	16#0	True	True	False		
bySDODataArray[35]	Byte	35.0	16#0	True	True	False		
bySDODataArray[36]	Byte	36.0	16#0	True	True	False		
bySDODataArray[37]	Byte	37.0	16#0	True	True	False		
bySDODataArray[38]	Byte	38.0	16#0	True	True	False		
bySDODataArray[39]	Byte	39.0	16#0	True	True	False		
bySDODataArray[40]	Byte	40.0	16#0	True	True	False		
bySDODataArray[41]	Byte	41.0	16#0	True	True	False		
bySDODataArray[42]	Byte	42.0	16#0	True	True	False		
bySDODataArray[43]	Byte	43.0	16#0	True	True	False		
bySDODataArray[44]	Byte	44.0	16#0	True	True	False		
bySDODataArray[45]	Byte	45.0	16#0	True	True	False		
bySDODataArray[46]	Byte	46.0	16#0	True	True	False		
bySDODataArray[47]	Byte	47.0	16#0	True	True	False		
bySDODataArray[48]	Byte	48.0	16#0	True	True	False		





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



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

Totally Integrated Automation Portal		
0277	#TG_400.Send.STW1.%X12 := TRUE; //turn on real time jogging mode	
0278	#bSWEnabled := TRUE;	
0279		
0280	// Do a Stop	
0281	IF #iStop THEN	
0282	#TG_400.Send.STW1.%X4 := FALSE;     // Stop active motion task	
0283	#TG_400.Send.STW1.%X5 := FALSE;     // Stop active motion task	
0284	#TG_400.Send.STW1.%X8 := FALSE;     // Stop jog 1	
0285	#TG_400.Send.STW1.%X9 := FALSE;     // Stop jog 2	
0286	#TG_400.Send.STW1.%X11 := FALSE;    // Stop homeing	
0287	//#TG_400.Send.STW1.%X12 := FALSE; // Stop real time jogging mode	
0288	ELSE	
0289		
0290	// Do home	
0291	IF #iStart_Homing AND NOT #bHomeStarted THEN	
0292	#TG_400.Send.HOME_DIST := #iCmdPos.%W1;	
0293	#TG_400.Send.HOME_DIST_1 := #iCmdPos.%W0;	
0294	#TG_400.Send.STW1.%X11 := TRUE;	
0295	#wHomingStartPoint := #TG_400.Receive.ZSW2;	
0296	#bHomeStarted := TRUE;	
0297	ELSE	
0298	IF #TG_400.Receive.ZSW1.%X11 AND #TG_400.Receive.ZSW1.%X13 AND (ABS(#wHoming-	
0299	StartPoint - #TG_400.Receive.ZSW2) > 5) THEN	
0300	#TG_400.Send.STW1.%X11 := FALSE;	
0301	END_IF;	
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	#TG_400.Send.STW1.%X6 := FALSE; //turn off the start bit	
0338	END_IF;	
0339	// Move	
0340	IF #iStart_Move AND NOT #bStartMove THEN	
0341	#TG_400.Send.MDI_MOD := #iMT_MoveType;     // 1 = absolute, 0 = relative	
0342	#TG_400.Send.SATZANW := #iMTNum;     // 0x8000 = direct motion task,	
0343	otherwise select a motion task from the AKD	
0344	#TG_400.Send.MDI_TARPOS := #iCmdPos.%W1;	
0345	#TG_400.Send.MDI_TARPOS_1 := #iCmdPos.%W0;	
0346	#TG_400.Send.MDI_VELOCITY := #iCmdVel.%W1;	
0347	#TG_400.Send.MDI_VELOCITY_1 := #iCmdVel.%W0;	
0348	#TG_400.Send.MDI_ACC := #iCmdAcc;	
0349	#TG_400.Send.MDI_DEC := #iCmdDec;	
0350	#TG_400.Send.STW1.%X4 := TRUE;     // Do not reject traversing task	
0351	#TG_400.Send.STW1.%X5 := TRUE;     // No intermediate stop	
0352	#TG_400.Send.STW1.%X6 := TRUE;     // Start the move	
0353		
0354	#bStartMove := TRUE; // the process to start the move has been started	
0355	but not completed	
0356	#bStartMoveDone := FALSE; // the process to start the move has not	
0357	been completed	
	#wStartMove := #TG_400.Receive.ZSW2;	
	END_IF; //If Start move	
	END_IF; //If Start Jog	
	END_IF; //If Start Homing	
	END_IF; // If Stop	

Totally Integrated Automation Portal		
0358	END_IF; // If ZSW1 bit2 Operation Enabled	
0359	END_IF; // If ZSW1 bit1 Switched On	
0360	END_IF; // If ZSW1 bit0 Ready for Switch On	
0361	END_IF; // If ZSW1 bit6 Switch-On Inhibited	
0362		
0363	// If the axis gets disabled all the corresponding motion signals have to be reset	
0364	ELSE	
0365	#bSWEnabled := FALSE;	
0366		
0367	#TG_400.Send.STW1.%X0 := FALSE;	
0368	#TG_400.Send.STW1.%X1 := FALSE;	
0369	#TG_400.Send.STW1.%X2 := FALSE;	
0370	#TG_400.Send.STW1.%X3 := FALSE;	
0371	#TG_400.Send.STW1.%X4 := FALSE;	
0372	#TG_400.Send.STW1.%X5 := FALSE;	
0373	#TG_400.Send.STW1.%X6 := FALSE;	
0374	#TG_400.Send.STW1.%X8 := FALSE;	
0375	#TG_400.Send.STW1.%X9 := FALSE;	
0376	#TG_400.Send.STW1.%X11 := FALSE;	
0377	#TG_400.Send.STW1.%X12 := FALSE;	
0378	END_IF; //If Enable	
0379	END_IF; // ZSW1 bit9 PLC has control	
0380	ELSE	
0381	#bPNWriteError := TRUE;	
0382	END_IF; //If #wPNWriteStatus = 0	
0383		
0384	// Read drive faults and warning on occurens	
0385	IF #oFault THEN	
0386	// If alarm is not read yet: start read	
0387	IF NOT #wSDOStatus.%X6 AND NOT #bSDOStatusReadActive THEN	
0388	#wSDOStatus := 64;	
0389	#wSDOAddress := 16#09AD; // Read PNU 2477 (DRV.FAULT1)	
0390	#bSDOStatusReadActive := TRUE;	
0391	END_IF;	
0392		
0393	// If warning is not read yet: start read	
0394	ELSIF #oWarning THEN	
0395	IF NOT #wSDOStatus.%X5 AND NOT #bSDOStatusReadActive THEN	
0396	#wSDOStatus := 32;	
0397	#wSDOAddress := 16#0AE7; // Read PNU2791 (DRV.WARNING1)	
0398	#bSDOStatusReadActive := TRUE;	
0399	END_IF;	
0400	ELSE	
0401	IF NOT #bSDOStatusReadActive THEN	
0402	#wSDOStatus := 16;	
0403	END_IF;	
0404	END_IF; //IF #oFault	
0405		
0406	// start the reading signal	
0407	IF #wSDOStatus >= 32 THEN	
0408	// Always clean out old values first	
0409	IF #bSDOWriteBusy THEN	
0410	#bSDOReq := FALSE;	
0411	ELSE	
0412	FILL_BLK(IN := 0,	
0413	COUNT := 64,	
0414	OUT => #bySDODdataArray[0]);	
0415	#bSDOReq := TRUE;	
0416	#bSDOWriteDone := FALSE;	
0417	END_IF;	
0418	// Start the read cylce to the corrspoding PNU	
0419	IF #wSDOStatus = 32 OR #wSDOStatus = 64 THEN	
0420	#bySDODdataArray[0] := #wSDOStatus.%B0;	
0421	#bySDODdataArray[1] := 16#01;	
0422	#bySDODdataArray[2] := 16#00;	
0423	#bySDODdataArray[3] := 16#01;	
0424		
0425	#bySDODdataArray[4] := 16#10;	
0426	#bySDODdataArray[5] := 16#01;	
0427	#bySDODdataArray[6] := #wSDOAddress.%B1;	
0428	#bySDODdataArray[7] := #wSDOAddress.%B0;	
0429	#bySDODdataArray[8] := 16#00;	
0430	#bySDODdataArray[9] := 16#00;	
0431		
0432	#SDOWrite(REQ := #bSDOReq,	
0433	ID := #iID,	
0434	INDEX := 47,	
0435	LEN := 14,	
0436	BUSY => #bSDOWriteBusy,	
0437	DONE => #bSDOWriteDone,	
0438	ERROR => #bSDOWriteError,	
0439	RECORD := #bySDODdataArray);	
0440	IF #bSDOWriteDone THEN	
0441	#wSDOStatus := #wSDOStatus + 1;	
0442	END_IF;	
0443	END_IF; //IF #wSDOStatus = 32 OR #wSDOStatus = 64	
0444		
0445	// read the value back	



```
0446     IF #wSDOStatus = 33 OR #wSDOStatus = 65 THEN
0447         #SDORead(REQ := #bSDOReq,
0448                 ID := #iID,
0449                 INDEX := 47,
0450                 MLEN := 14,
0451                 BUSY => #bSDOReadBusy,
0452                 VALID => #bSDOReadDone,
0453                 ERROR => #bSDOReadError,
0454                 RECORD := #bySDODataArray);
0455     IF #bSDOReadDone THEN
0456         IF #bySDODataArray[4] = 66 THEN // if the read signal is a Word use Byte 6&7
0457             #wSDOReadValue.%B0 := #bySDODataArray[7];
0458             #wSDOReadValue.%B1 := #bySDODataArray[6];
0459         ELSE // if the read signal is a DWord use Byte 8&9
0460             #wSDOReadValue.%B0 := #bySDODataArray[9];
0461             #wSDOReadValue.%B1 := #bySDODataArray[8];
0462         END_IF;
0463         IF #wSDOReadValue = 0 THEN // If the read value is zero repeat the read cycle
0464             #wSDOStatus := #wSDOStatus - 1;
0465         ELSE
0466             #wSDOStatus := #wSDOStatus + 1;
0467             #bSDOStatusReadActive := FALSE;
0468         END_IF;
0469     END_IF; //IF #bSDOReadDone
0470 END_IF; //IF #wSDOStatus = 33 OR #wSDOStatus = 65
0471 END_IF; //IF #wSDOStatus >= 32
0472 END_IF; //IF NOT #iInit AND #wSDOStatus > 15
0473
0474 // check for rising edge for the input parameters
0475 IF NOT #iReset AND #bResetDone THEN
0476     #bResetDone := FALSE;
0477 END_IF;
0478 IF NOT #iStart_Homing AND #bHomeStarted THEN
0479     #bHomeStarted := FALSE;
0480 END_IF;
0481 IF NOT #iStart_Move AND #bStartMove AND #bStartMoveDone AND NOT #TG_400.Receive.ZSW1.%X12 THEN
0482     #bStartMove := FALSE;
0483     #bStartMoveDone := FALSE;
0484 END_IF;
0485
0486 // fault detection/handling within the bloc
0487 IF #bPNReadError OR #bPNWriteError OR #bSDOReadError OR #bSDOWriteError THEN
0488     #oError := TRUE;
0489     IF #bPNReadError THEN
0490         #oStatus := #wPNReadStatus; //indicates internal FB error
0491     END_IF;
0492     IF #bPNWriteError THEN
0493         #oStatus := #wPNWriteStatus; //indicates internal FB error
0494     END_IF;
0495 ELSIF #wSDOStatus > 15 THEN
0496     #oError := FALSE;
0497 END_IF;
0498
```

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

Totally Integrated Automation Portal		
Symbol	Address	TypeComment
#iStart_Jog		Boolstart jog
#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
#oInitDone		Boolinitialization status - 0 while configuring TG400, 1= done
#oInPos		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