1. Application description

This application presents an example on how to restore drive operation after a fault state.

Default behavior in case of an error:

- Switch off green LED
- Switch on red LED
- Cutoff drive PWM outputs and disable the controllers
- Terminate the execution of the TML program.

Proposed recovery sequence:

- Switch off the green LED
- Switch on the red LED
- Cutoff drive PWM outputs
- Wait until the error condition disappears
- Program the motor to hold its current position
- Re-activate the drive PWM outputs
- Switch off the red LED and switch on the green LED
- Resume the main program and process the commands received from the master.

The application uses the software protection interrupt, which monitors the following software protections:

- Over current;
- Over temperature drive (where a temperature sensor is available);
- Over voltage;
- Under voltage;

An easiest way to evaluate this application is to trigger an over-voltage. This can be done by decreasing the over-voltage protection threshold value under the supply voltage value. The parameter that stores the over voltage protection threshold value is called "UMAXPROT". It can be modified online from the Command Interpreter window in EasyMotion Studio. For details, see the chapter 4.3.

2. Application flow chart



Figure 1. Main application structure



Figure 2. Software protections interrupt routine structure

3. EasyMotion Studio implementation



Figure 3. Main section of the TML program

Project	1 2		3 4 5 6 7 8	
Image: Set in the set in	Project			×
Image: Sector the operation Interrupt int2 Image: Sector the operation Interrupt int2 Image: Sector the operation Image: Sector the operation Image: Sector the operation Ima	A A M A SEIN OF A W	Τc		V.A.
Set Mill Application Image: Set Setup Image: Setup		-		_ @
B Bit Action the operation S Stup Image: Stup B Image: Stup Control (2) = condob, fist I/O ine 2 to Low Image: Stup Stup Image: Stup Control (2) = condob, fist I/O ine 2 to Low Image: Stup Stup Image: Stup <t< td=""><td></td><td>H</td><td>Interrupt Int2</td><td>l</td></t<>		H	Interrupt Int2	l
S Setup S Detro ELDOVE S Motion S OUT(2) = 0x0000, //Set I/D line 2 to Low Image: Setup S Detro ELDOVE S OUT(2) = 0x0000, //Set I/D line 3 to High Image: Setup S Detro ELDOVE S OUT(2) = 0x0000, //Set I/D line 3 to High Image: Setup S Detro ELDOVE P Descrivate the control loops and the power stage. The motor will stop freely. WARNING: Disabling the next instruction may damage your drive / motor Y Image: Set the "err, status" user variable to 1, to mark that the error status has occured and set this variable to the host. Image: Set the "err, status" user variable to 1, to mark that the error status has occured and set this variable to the host. Image: Set the "err, status" user variable to 1, to mark that the error status has occured and set this variable to the host. Image: Set the "err, status" user variable to 1, to mark that the error status has occured and set this variable to the host. Image: Set the "error, attatus" user variable error, status has occured and set this variable to the host. Image: Set the "error, attatus" user variable error, status has occured and set this variable to the host. Image: Set the "error, attatus" user variable error, status has occured and set this variable to the host. Image: Set the "error, attatus" user variable error, status has occured and set this variable to the host. Image: Set the "error, attatus" user variable error, status has occured and set this line.	E E Restore the operation		Default actions when a software protection is triggered	中国
Motion <	S Setup	-	ETTOP LED ON	
Marken Modes Functions CAM Tables CAM Tables CAM Tables Sectivate protections Interrupts CAM Tables CAM Tables WARNING: Disabling the next instruction may damage your drive / motor! Variable status 'user variable to 1, to mark that the error status has occurred and set this variable to the host. Final Sectivate protections WARNING: Disabling the next instruction may damage your drive / motor! Variable status 'user variable to 1, to mark that the error status has occurred and set this variable to the host. Final Sectivate the control loops and PVM outputs Section	M Motion	5		+
Functions Control = Set Worksons, Detroy, Det	Homing Modes	-	Ready LCD OF	Ť
CAM Tables CAM Tables CAM Tables WARNING: Disabling the next instruction may damage your drive / motor V VARNING: Disabling the next instruction may damage your drive / motor V CAM Tables Q AvisioPF; // Deactivate the control loops and PVM outputs E Set the "err, status = 1; T isstuar user variable to 1, to mark that the error status has occured and set this variable to the host. 6 6 7 7 8500 error, status, "User variable to 1, to mark that the error status has occured and set this variable to the host. 6 7 7 7 8500 error, status, "User variable to 1, to mark that the error status has occured and set this variable to the host. 6 7 7 9 7 7 9 10	E Functions	5	Out(s) = 0x0000, //set voline s to high	×
 AXISOFF, // Deactivate the control loops and PVM outputs AXISOFF, // Deactivate the control loops and PVM outputs Set the 'err_status' user variable to 1, to mark that the error status has occured and set this variable to the host. MASTERID = 4001; // Set host address to 255 SENU err_status, // Bend to host contents of variable err_status Africe excuting this instruction, the drive / motor will enter the FAULT status. This action will also stop the execution of the TML program NOTE: To implement a custom error recovery, you can insert your own error recovery sequence before this line, vi Wait until the error cause is gone by testing the Protections Control Register (PCR) Wait until the error cause is gone by testing the Protections Control Register (PCR) Wait until the error cause is gone by testing the Protections Control Register (PCR) Wait until the error cause is gone by testing the Protections Control Register (PCR) Wait until the error cause is gone by testing the Protections Control Register (PCR) Wait until the error cause is gone by testing the Protections Control Register (PCR) Wait until the error cause is gone by testing the error cause is gone if VarAux via the error gone if VarAux is 0 Wait 0.5 second Wait 0.5 second Wait 0.5 second Gent the motor tho tait a current position Position profile CACC = 0.5863/acceleration rate = 2000[radis*2] CACC = 0.5863/acceleration rate = 2000[radis*2] CACC = 0.5863/acceleration rate = 2000[radis*2] CACC = 0.6863/acceleration rate = 2000[radis*2] CASDN // Activate the Control loops and PVMI outputs Error LED OFF OUT(2) = 0x0004,	Interrupts Interrupts Int2 - Software protections Can Tables		Deactivate the control loops and the power stage. The motor will stop freely, WARNING: Disabling the next instruction may damage your drive / motor!	
 Set the "err_status" user variable to 1, to mark that the error status has occured and set this variable to the host. Get the "err_status" user variable of 1, to mark that the error status has occured and set this variable to the host. MASTERD - 4061; // Set host address to 255 SEND err_status; // Send to host contents of variable err_status <i>p</i>'' Trigger the FAULT status by setting b8 3 of SRL register. After executing this instruction, the drive / motor will enter the FAULT status. This action will also stop the execution of the TML program NOTE: To implement a custom error recovery, you can insert your own error recovery sequence before this line. Y'' VarAux = PCR: SRB VarAux, 0xFF00, 0x0; //Set VarAux using AND mask 0xFF00 and OR mask 0x0000 GotTO wat_error_gone. (VarAux, NEC), //Branch to wat_error_gone if VarAux + 0 Watt 0.5 second Y ar Solo: Watt S second Set the motor to hold its current position //Position profile CACC = 0.5363/(acceleration rate = 2000(rad/s*2) CACC =	-	-	7 AVEORE // Deathingto the control loans and DMM autouts	
□ Set the "m_status" user variable to 1, to make the error satus has occurred and set this Variable to the float. 6 err_status Yet host address to 255 7 SEND err_status, // Set host address to 255 SEND err_status, // Set host address to 256 7 Trigger the FAULT status by setting b43 of SRL register. After executing this instruction, the drive / motion will enter the FAULT status. This action will also stop the execution of the TML program NOTE: To implement a custom error recovery, you can insert your own error recovery sequence before this line. 9 Wat until the error cause is gone by testing the Protections Control Register (PCR) 4 wat_error_pone. (Pde fine label wat_error_gone if Variaux is 0 6 Variaux = PCR; Set off Variaux is 0.000 6 Status 0.5 second Status; NED, //Branch to wat_error_gone if Variaux is 0 9 Wat 0.5 second Status; Status; Variaux, NED, //Branch to wat_error_gone if Variaux is 0 9 Wat 0.5 second Status; Status 0.5 second Status; Status; Variaux, Status; Variaux, Status; Variaux, Status; Variaux; Status; Variaux; Status; Variau;		12	AVADUTE, II beautivate the control loops and MVM outputs	N
6 ortstatus = 1, 7 MASTERD = 4081; // Set host address to 255 7 SEDD err_status, // Send to host contents of variable err_status /* Trigger the FAULT status by setting bit 3 of SRL register. After executing this instruction, the drive / motor will enter the FAULT status. This action will also stop the execution of the TML program? NOTE: To implement a custom error recovery, you can insert your own error recovery sequence before this line. 7 Wat until the error cause is gone by testing the Protections Control Register (PCR) 4 wat_error_gone: //Define label wat_error_gone 6 Varkux = PCR; 6 Status = error_gone, VarAux, NEQ; //Branch to wat_error_gone if VarAux != 0 1 Wat 0.5 second 1 //Define event: After a wait time equal with value 0.5 s 1 //Define event: After a wait time equal with value 0.5 s 1 (CACC = 0.8566)/acceleration rate = 2000[rad/s*2] 1 CACC = 0.8566/acceleration rate = 2000[rad/s*2] 1 CACC = 0.8566/acceleration rate = 2000[rad/s*2] 2 CAXD01, // Activate the control lops and PWM outputs 2 AXXD01, // Activate the PWM outputs and "unfreeze" the controllers) 2 AXD01, // Activate the control lops and PWM outputs		-	Joe me en_status user variable to 1, to mark that the error status has occured and set (his variable to the host, arr status = 1	R
7 SRID GPT Status, // Set individual to bad contents of variable err_status /* Trigger the FAULT status by setting bt 3 of SRL register. After executing this instruction, the drive / motor will enter the FAULT status. This action will also stop the execution of the TML program! NOTE: To implement a custom error recovery, you can insert your own error recovery sequence before this line. 7 Wat unlit the error cause is gone by testing the Protections Control Register (PCR) 4 wat_error_gone. //Define label wat_error_gone 6 Varkux = PCR; 6 SRD Varkux, DEFPO, bx0. //Set VarAux using AND mask 0x/FP0 and OR mask 0x0000 4 Oor wat_error_gone, VarAux, NEO, //Branch to wat_error_gone if VarAux != 0 9 Wat 0.5 second 7 // Define event: After a wait time equal with value 0.5 s 8 Set the motor to hold its current position 7 //Bestation profile CACC = 0.83683//acceleration rate = 2000[rads*2] CSPD = 5.1//plosition command = 10[U] CPR = 0.1//position command = 10[U]		6	CIL_SIGUS = 1, MACTEDD = 4081: // Set host address to 255	
/* Trigger the FAULT status by setting bit 3 of SRL register. After executing this instruction, the drive / motor will enter the FAULT status. This action will also stop the execution of the TML program NOTE: To implement a custom error recovery, you can insert your own error recovery sequence before this line. /* B Wait util the error cause is gone by testing the Protections Control Register (PCR) 4 wait_error_gone. //Define label wait_error_gone 6 VarAux, a PCR; 6 SRB VarAux, oxFP00, 0x0. //Set VarAux using AND mask 0xFF00 and OR mask 0x0000 4 OOT wait_error_gone, VarAux, NEQ, //Branch to wait_error_gone if VarAux is 0 9 Wait 0.5 second /// Define event: After a wait time equal with value 0.5 s 18T 500L: WAIT, // Wait until the event occurs 9 Set the motor to hold its current position // Position profile CACC = 0.83683/ucceleration rate = 2000[rad/s*2] CSD = 0.L//position command is relative unODE PR U/D PR WODE // if vacuute the PVMI outputs and "unfreeze" the controlers) 2 AxISON; // Activate the control loops and PVMI outputs 1 Enable the motor (activate the PVMI outputs and "unfreeze" the controlers) 2 AxISON; // Activate run erotherolino pasing and PVMI outputs <t< td=""><td></td><td>7</td><td>SEND err_status; // Sen to host contents of variable err_status</td><td>Þ</td></t<>		7	SEND err_status; // Sen to host contents of variable err_status	Þ
V Wat until the error cause is gone by testing the Protections Control Register (PCR) Wat_error_gone: //Define label wat_error_gone 6 VarAux = PCR; 6 SRB VarAux, 0xFF00, 0x0; //Set VarAux using AND mask 0xFF00 and OR mask 0x0000 4 GOTO wat_error_gone, VarAux, NEQ; //Branch to wat_error_gone if VarAux != 0 9 Wat 0.5 second // Define event : After a wat time equal with value 0.5 s // To Erine event : After a wat time equal with value 0.5 s // To Stion profile // CAC = 0.85633//acceleration rate = 2000[rad/s*2] CSPD = 5.//slew speed = 160[rpm] 1 CPS = 0.1//position command is relative MODE PP; // UPD; // execute immediate III Enable the motor (activate the PWM outputs and "unfreeze" the controllers) 2 AXISON; // Activate the control loops and PVM outputs B Enror LED OFF 0.07(2) = 0x0004; //Set V0 line 2 to High III Ready LED ON Encord for a four of a four o			/* Trigger the FAULT status by setting bit 3 of SRL register. After executing this instruction, the drive / motor will enter the FAULT status. This action will also stop the execution of the TML program? NOTE: To implement a custom error recovery, you can insert your own error recovery sequence before this line.	
Image: Strate of the state		⊢	1/	
4 WarAux = PCR; 6 VarAux = PCR; 7 SRB VarAux, 0xFF00, 0x0; //Set VarAux using AND mask 0xFF00 and OR mask 0x0000 4 GOTO wat_error_gone, VarAux, NEQ; //Branch to wat_error_gone if VarAux != 0 2 Wat 0.5 second 3 //D befine event: After a wait time equal with value 0.5 s 3 //R 500L; WATI, // Wat until the event occurs 2 Set the motor to hold its current position //Position profile CACC = 0.83663//acceleration rate = 2000[rad/s*2] CAPD S = 0L://position command = 0[U] CPR: //position command is relative MODE PP UPD; // execute immediate 2 Enable the motor (activate the PWM outputs and "unfreeze" the controllers) 2 AXISON; // Activate the control loops and PWM outputs 3 DUT(2) = 0x0004; //Set VO line 2 to High 3 Ready LED ON		-	wait error none //Define lahel wait error none	
0 StRE VarAux, 0xFF00, 0x0; //Set VarAux using AND mask 0xFF00 and OR mask 0x0000 4 GOTO wat_error_gone, VarAux, NEQ; //Branch to wat_error_gone if VarAux != 0 B Wat 0.5 second // Define event : After a wait time equal with value 0.5 s 3 #T 500L; Wait until the event occurs B Set the motor to hold its current position //Position profile CACC = 0.83683//acceleration rate = 2000[rad/s^2] CSPD = 5: //slew speed = 150[rpm] CPC //position command = 0[U] CPR: //position command is relative MODE PP; UPD; // execute immediate Enable the motor (activate the PWM outputs and "unfreeze" the controllers) 2 AXISON; // Activate the control loops and PWM outputs 5 OUT(2) = 0x0000; //Set V0 line 2 to High B Ready LED ON Stown, //Set V0 line 3 to Low		4	Var_one_jen: house here yes and jene yes and the second se	
0 GOTO wat_error_gone, VarAux, NEQ; //Branch to wat_error_gone if VarAux != 0 4 GOTO wat_error_gone, VarAux, NEQ; //Branch to wat_error_gone if VarAux != 0 5 Wat 0.5 second 7 // Define event: After a wait time equal with value 0.5 s 8 IRT 500L; WART; // Wait until the event occurs 5 Set the motor to hold its current position 7 //Position profile CACC = 0.53663//acceleration rate = 2000[rad/s^2] CSPD = 5.://slew speed = 150[rpm] CPOS: //position command is relative MODE PP, UPD; // execute immediate 8 Enable the motor control loops and PWM outputs 9 Error LED OFF 5 OUT(2) = 0x0000; //Set VO line 2 to High 9 Ready LED ON 5 Out(2) = 0x0000; //Set VO line 3 to Low 5 Ready LED ON 6 Bernor LED ON		0	SPR VarAux: 0xFE00_0x0: //Set VarAux using AND mask 0xFE00 and OR mask 0x0000	
Image: Second Image: Second Image: Image: Image: Second Image: Second Image: Image: Image: Image: Image: Image: Second			GOTO wat error gone. VarAux NEG: //Branch to wait error gone if VarAux != 0	
// Define event: After a wait time equal with value 0.5 s 3 IRT 500L; WATT; // Wait until the event occurs ID Set the motor to hold its current position //Position profile CACC = 0.63663//acceleration rate = 2000[rad/s*2] CSPD = 5.1/slew speed = 150[rpm] CPOS = 0L://position command is relative MODE PP; UPD; // execute immediate ID Enable the motor (activate the PWM outputs and "unfreeze" the controllers) AXISON: // Activate the control loops and PWM outputs SOUT(2) = 0x0000; //Set VO line 2 to High ID Ready LED ON SOUT(2) = 0x0000; //Set VO line 3 to Low ID Ready LED ON		+	E Wait 0.5 second	
B Set the motor to hold its current position //Position profile CACC = 0.63663//acceleration rate = 2000[rad/s^2] CSPD = 5.:/slew speed = 150[rpm] CPOS = 0L.://position command = 0[U] CPR:://position command is relative MODE PP; UPD; // execute immediate B Enable the motor (activate the PWM outputs and "unfreeze" the controllers) AXISON: // Activate the control loops and PWM outputs B Error LED OFF 5 OUT(2) = 0x0000; //Set VO line 2 to High B Ready LED ON 5 OUT(2) = 0x0000; //Set VO line 3 to Low CH Reset the settive protections interruet fine and fault status		3	// Define event : After a wait time equal with value 0.5 s IRT 500L; WAIT; // Wait until the event occurs	
//Position profile CACC = 0.63663//acceleration rate = 2000[rad/s*2] CSPD = 5.1/slew speed = 150[rpm] CPOS = 0L.//position command = 0[U] CPR, //position command is relative MODE PP, UPD; // execute immediate E Enable the motor (activate the PVM outputs and "unfreeze" the controllers) AXISON, // Activate the control loops and PWM outputs SOUT(2) = 0x0004; //Set VO line 2 to High E Ready LED ON SOUT(2) = 0x0000; //Set VO line 3 to Low E Ready LED ON			Set the motor to hold its current position	
Enable the motor (activate the PWM outputs and "unfreeze" the controllers) AXISON, // Activate the control loops and PWM outputs Fror LED OFF OUT(2) = 0x0004; //Set VO line 2 to High Ready LED ON OUT(3) = 0x0000; //Set VO line 3 to Low OUT(3) = 0x0000; //Set VO line 3 to Low Control the software protections interrupt flap and fault status		1	//Position profile CACC = 0.63863://acceleration rate = 2000[rad/s^2] CSPD = 5:://slew speed = 150[rpm] CPOS = 01:://position command = 0[IU] CPR: //position command is relative MODE PP; UPD; // execute immediate	
AXISON; // Activate the control loops and PWM outputs Berror LED OFF OUT(2) = 0x0004; //Set VO line 2 to High Bready LED ON OUT(3) = 0x0000; //Set VO line 3 to Low DReset the software protections interrupt flee and fault status			Enable the motor (activate the PWM outputs and "unfreeze" the controllers)	
Error LED OFF OUT(2) = 0x0004; //Set VO line 2 to High Ready LED ON OUT(3) = 0x0000; //Set VO line 3 to Low OUT(3) = 0x0000; //Set VO line 3 to Low		2	AXISON; // Activate the control loops and PWM outputs	
5 OUT(2) = 0x0004; //Set VO line 2 to High Control = Ready LED ON Control = 0x0000; //Set VO line 3 to Low 5 OUT(3) = 0x0000; //Set VO line 3 to Low Control = Ready LED ON Control = 0x0000; //Set VO line 3 to Low			Error LED OFF	
Ready LED ON OUT(3) = 0x0000; //Set VO line 3 to Low Reset the software protections interrupt flap and fault status		5	OUT(2) = 0x0004; //Set VO line 2 to High	
5 OUT(3) = 0x0000; //Set VO line 3 to Low		F	Ready LED ON	
Reset the software protections interrupt flag and fault status		5	OUT(3) = 0x0000; //Set VO line 3 to Low	
		-	Reset the software protections interrupt flag and fault status	
8 FAULTR: //Reset FAULT status		8	FAULTR; //Reset FAULT status	
Set the "err_status" user variable to 0, to mark that the error state has disappeared and set this variable to the host.			Set the "err_status" user variable to 0, to mark that the error state has disappeared and set this variable to the host.	
6 err_status = 0;		6	err_status = 0;	
MASTERID = 4081; // Set host address to 255		7	MASTERD = 4081; // Set host address to 255	
SEND err_status; // Send to host contents of variable err_status		1	SEND err_status; // Send to host contents of variable err_status	
4 ⊟ Set the "flag" user variable to 1, to force the cod to return to master command loop.		4	Set the "flag" user variable to 1, to force the cod to return to master command loop.	
0 ing = 1,		10	nay = 1;	

Figure 4. Software protections interrupt service routine

4. Detailed description of the EasyMotion Studio implementation

4.1 Motion section

The code sequences from the "Motion" section were generated using the buttons marked with 1 to 5 in Figure 2. Clicking on those buttons the following programming dialogs will open.

• The "Miscellaneous" dialogue (5) allows defining user variables (integer - 16 bit, fixed - 32 bit or long -32 bit).

Miscellaneous	?	Х	Miscellaneous ? X
	•		Define variable named cmd of type int
Reset FAULT status			C Reset FAULT status
C Insert END instruction			C Insert END instruction
C Insert <u>N</u> OP instruction			C Insert <u>N</u> OP instruction
Serial communication C Change baudrate to			Serial communication C Change baudrate to CAN communication
C Change baudrate to			C Change baudrate to
C Insert ENDINIT instruction			C Insert ENDINIT instruction
C Save actual setup data in the EEPROM			Save actual setup data in the EEPROM
OK Cancel Help			OK Cancel Help

Figure 5. How to define variables

• The "Assignment and Data Transfer – 16 bit Integer Data" dialogue (6) helps to assign a value to a 16-bit integer variable. This way a variable can be initialized or its value can be modified.

Assignment & Data Transfer - 16 bit Integer Data	?	×	Assignment & Data Transfer - 16 bit Integer Data	?	Х
Set 16-bit variable VarAux					
With value / <u>1</u> 6 bit variable / label			With value / <u>1</u> 6 bit variable / label		
C With C data memory contents, located at address set in pointer variable C E2ROM Then increment the pointer variable			C With C grogram C E2ROM then increment the pointer variable		
C With C low C high part of 32-bit variable			C With C jow part of 32-bit variable		
C With the inverse (-) of variable			C With the inverse (-) of variable		
C Lising AND mask h and OR mask h			C Lising AND mask h and OR mask h		
C With checksum C data correction of data located in C rorgram C E2ROM and	h h		C With checksum of data located in C program C E2ROM and	h h	
C Set C data C program C E2ROM C E2ROM C E2ROM C E2ROM C then increment the pointer variable			C Set C data C program C E2ROM C E2ROM C E2ROM C expression C expressi		
OK Cancel Help			OK Cancel Help		

Figure 6. How to set a value for a variable

• The "Jumps and Function Calls" dialogue (4) allows controlling the TML program flow through unconditional or conditional jumps and unconditional, conditional or cancelable calls of TML functions.

In this case this dialog was used to create the "MasterCommand" loop where the program waits for the master to start the motion sequence, by setting the "cmd" user variable to a value different than 0.

Jumps and Function Calls			?	×
C <u>G</u> oto at address, label C <u>C</u> all or C Cancelable Call if variable C RETurn from function C RETurn from interrupt C Abort cancelable call	is	C EQ C LT C GI	C <u>N</u> EQ C LE <u>Q</u> C <u>G</u> EQ	than O
Insert label named MasterCommand OK Cancel Help				
Jumps and Function Calls			?	×
Image: Call gamma and address, label or at address, label or at address set in variable MasterCommand Image: Cancelable Call Image: Call gamma address set in variable Image: Call gamma address set in variable Image: Call gamma address set in variable Image: Call gamma address set in variable Image: Call gamma address set in variable Image: Call gamma address gamma address set in variable Image: Call gamma address gamma	is	© <u>E</u> Q ○ <u>L</u> T ○ G <u>I</u>	C <u>N</u> EQ C LE <u>Q</u> C <u>G</u> EQ	than 0
OK Cancel Help				

Figure 7. How to implement a TML loop

• The "Motion – Trapezoidal Profiles" dialogue (1) allows to program a position or speed profile with a trapezoidal shape of the speed, due to a limited acceleration.

Motion - Trapezoidal Profi	les					? ×
	• Position • Speed	⊂ <u>A</u> bsolu ⊙ <u>R</u> elati	ute ve 🗆 Addi	itive	Execute	Then wait until motion is completed Change Event
Acceleration rate =	1000		rad/s^2	-	1	EaltEvent
Slew speed =	100		rpm	•		
Position increment =	5		rot	-		
⊙ Gene value ⊖ Gene value	rate new trajectory s of position and sp rate new trajectory s of load/motor pos	starting from eed referent starting from sition and sp	m actual nce m actual peed		C Setup motion don't start exe	data, but ecution
	OK	C	ancel	H	elp	

Figure 8. How to set a motion profile

This dialog was used to create the motion profiles inside the "MachineProgram" loop.

• The "Events" dialogue (2) allows to define events. An event is a programmable condition, which once set, is monitored for occurrence.

The following actions can be connected to an event:

- stop the motion when the event occurs;
- wait for the programmed event to occur.

Events				?	×
Set Event Change Eve	After a wait tir	me equal with value	: 0.5 s		^
<u>E</u> dit Event					~
🔲 Stop the mo	tion when the eve	ent occurs			
🔽 🔟 ait until th	e event occurs				
E <u>x</u> it from	the wait loop afte	r a time equal with			~
	ОК	Cancel	Help		

Figure 9. How to create an event

The "Events" dialog is used to insert a delay of 0.5 s between the two motion profiles.

4.2 Software Protections Interrupt

The TML interrupts are special functions that are continuously monitored by the drive firmware. When a TML interrupt occurs, the main TML program execution is suspended and the TML code associated with the interrupt, called Interrupt Service Routine (in short ISR), is executed.

<u>Remark</u>: While an interrupt is active, the other interrupts are deactivated. It is recommended to keep the ISR as short as possible. If is not possible, then the other interrupts should be re-enabled using the "Interrupts Settings" dialogue.

Giobally disable The Interrupts	Enable/Dieable	TMI Interrunte	Reset previous TML	
TML Interrupts	Enable Disable		interrupt requests	
t0 - Enable input has changed	Г	Г	Г	
t1 - Short-circuit		Γ	Г	
t2 - Software protections	Γ		Г	Details
t3 - Control error				
t4 - Communication error				
t5 - Position wraparound				
t6 - LSP programmed transition detected				Details
t7 - LSN programmed transition detected				Details
t8 - Capture input transition detected				Details
t9 - Motion is completed / in velocity				
t10 - Time period has elapsed				Details
t11 - Event set has occurred			Γ	
t12 - Position trigger 14 change detected				

Figure 10. Interrupt Settings dialog

This application was implemented using the "Int2 - Software protections" interrupt, that was customized to contain the code that corresponds to the recovery sequence proposed in the first chapter of this document.

The code sequence inside the "Int2 - Software protections" interrupt was generated using the buttons marked with 1 to 8 in Figure 3. Clicking on those buttons the following programming dialogs will open.

• The "I/O" dialogue allows programming the following operations with the digital inputs and outputs:

- read and save the status of a digital input into a variable
- set low or high a digital output
- read and save the status of multiple digital inputs into a variable
- set multiple digital outputs according with an immediate value or the value of 16-bit variable

In this application, the "I/O" dialogue (1) is used to set LOW / HIGH the "Ready" and "Error" digital output, that are also associated to the green and respective red LED on the drive.

1/0 ×	1/0 ×
Single I/O Read an input Read input line into variable	Single I/D Read an input Read input line into variable
Set an output Set output line OUT2/Error status C low C high Set 1/0 line type Set as output	Set an output
C Read input(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 into variable	C Read input(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 into variable
C Set output(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 C as High C C C C C C C C C C C C C C C C C C C	C Set output(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 C as High C<
OK Cancel Help	OK Cancel Help

Figure 11. How to set output line status

• The "Motion - Motor Commands" dialogue (2) was used to deactivate the control loops and the power stage PWM output commands (AXISOFF) when the error has occurred.

Motion - Motor Commands	×
Activate the control loops and PWM outputs (AXISON) Deactivate the control loops and PWM outputs (AXISOFF) Stop motion with acceleration / deceleration set	
C Immediate update C Update on event	
C Set actual position to	
C Set quick stop deceleration rate to	-
OK Cancel Help	

Figure 12. How to deactivate the drive power stage

• The "Assignment and Data Transfer – 16 bit Integer Data" dialogue (6) was used to modify the value of the "err_status" and "flag" user variables.

Assignment & Data Transfer - 16 bit Integer Data	?	×	Assignment & Data Transfer - 16 bit Integer Data	?	×
C Set 16-bit variable er_status			Set 16-bit variable flag		
With value / <u>1</u> 6 bit variable / label			With value / <u>1</u> 6 bit variable / label		
C With C data C D grogram C E2ROM then increment the pointer variable			C With C gata C E2ROM C E2ROM		
C With C jow part of 32-bit variable			C With C jow C high		
○ <u>W</u> ith the inverse (·) of variable			C With the inverse (-) of variable		
C Lising AND mask h and OR mask h			C Using AND mask h and OR mask h		
C With checksum of data located in C program C E2ROM and	h h		C With checksum C data C program C E2ROM and	h h	
C Set C data C program C E2ROM OK C Gata C program C E2ROM C E2ROM C Cacle Help C E2ROM C Cacle Help C C Cacle C C Cacle C C C C C C C C C C C C C C C C C C C			C Set C data C program C E2ROM C E2ROM C C C C C C C C C C C C C C C C C C C		

Figure 13. How to set a value for a variable

• The "Jumps and Function Calls" dialogue (4) was used to create the "wait_error_gone" loop, where the program will stay as long as the "VarAux" variable will be different than 0.

Jumps and Function Calls	?	×
Image: Sector of Call of Call at address, label of at address set in variable of at address set in variable of if variable VarAux is C EQ C Cancelable Call Image: C RETurn from function C RETurn from interrupt C Abort cancelable call Image: C Insert label named: Image: OK		than O
Jumps and Function Calls	?	×
Image: Constraint of the second state of the second st	© NEQ ∩ LE <u>Q</u> ∩ <u>G</u> EQ	than 0
C Insert label named:		

Figure 14. How to implement a TML loop

• Inside the "wait_error_gone" loop the "Assignment and Data Transfer – 16-bit Integer Data" dialogue (6) was used, to apply an "AND" and "OR" mask to the "VarAux" user variable (copy of the PCR register). The mask isolates bits 15 to 8 in the "VarAux" variable, to check if the error state has disappeared.

Assignment & Dat	a Transfer - 16 bit Integer Data	? ×					
Get 16-bit ⊻	ariablej VarAux						
C With ve	alue / <u>1</u> 6 bit variable / label						
C With	C data memory contents, located at address set in pointer variable C E2ROM then increment the pointer variable	le					
C With	C jow C high						
⊂ <u>W</u> ith th	e inverse (-) of variable						
⊙ <u>U</u> sing A	AND mask FF00 h and OR mask 0000 h						
⊂ With cł of data	located in C data C program C E2ROM and C	h h					
C Set C C	C Set C data C program C E2ROM C between the pointer variable C between the pointer variable						
	OK Cancel Help						

Figure 15. How to apply an AND and OR mask to a 16-bits variable

<u>Remark</u>: PCR (Protections Control Register) is a 16-bit command and status register, containing the status information of the TML protections. A detailed description of the PCR register in presented in the EasyMotion Studio help topics (<u>link</u>).

• The "Events" dialogue (3) was used to hold the program execution for 0.5 s.

Events	?	×
Set Event After a wait time equal with value	l.5 s	^
<u>E</u> dit Event		\sim
Stop the motion when the event occurs		
✓ Wait until the event occurs		
\square Exit from the wait loop after a time equal with		-
0K Cancel	Help	

Figure 16. How to set a time event

• The "Motion – Trapezoidal Profiles" dialogue (1) was used this time to force the motor to hold the current position, after the error state disappears and the axis is re-enabled.

1otion - Trapezoidal Pro	files		? ×				
İZ.		C Absolute G Belative □ Additive	Execute Then wait until motion is completed Con event				
Acceleration rate =			Edit Event				
Slew speed =							
Position increment =	0	IU 💌					
Generate new trajectory starting from actual values of position and speed reference Generate new trajectory starting from actual Setup motion data, but							
values of load/motor position and speed							
	OK	Cancel	felp				

Figure 17. How to configure and start a position profile with CPOS = 0, using the TUM0 mode

• The "Motion - Motor Commands" dialogue (2) is used again to reactivate the drive PWM outputs and allow the position profile above to start being executed.

Motion - Motor Commands	×
 Activate the control loops and PWM outputs (AXISDN) Deactivate the control loops and PWM outputs (AXISDFF) Stop motion with acceleration / deceleration set 	
C Immediate update	
C Update on event	_
Set position: C Set actual position to C value C variable	
C Set quick stop deceleration rate to C value C variable	
OK Cancel Help	

Figure 18. How to activate the control loop

• The green LED (Ready – OUT3) is turned on and the red LED (Error – OUT2) is switched OFF, using the "I/O" dialogue.

1/0 ×	1/0 ×			
Single 1/D Read an input C Read input line into variable	Single I/D Read an input C Read input line into variable			
Set an output ⓒ Set output line DUT2/Error ▼ status ◯ low ⓒ high Set I/D line type Set as output	output Set an output output line OUT2/Error status Iow fine type Set as output Set I/D line type Set as output			
C Read input(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 into variable	C Read input(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 into variable			
C Set output(s) 15 14 13 12 11 10 9 7 6 5 4 3 2 1 0 C as High C <t< th=""><th>C Set output(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 C as High C <t< th=""></t<></th></t<>	C Set output(s) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 C as High C <t< th=""></t<>			

Figure 19. How to reset the output line status

• The fault status is reset using the "FAULTR;" TML instruction (see the "Miscellaneous" (8) dialog).

Miscellaneous	?	Х
C Define variable named of type	v	
Reset FAULT status		
C Insert END instruction C Insert <u>N</u> OP instruction		
Serial communication C Change baudrate to CAN communication C Change baudrate to		
C Insert ENDINIT instruction		
Save actual setup data in the EEPHUM OK Cancel Help		

Figure 20. How to reset fault status

<u>**Remark**</u>: The drive/motor will return to FAULT state if there are errors when the FAULTR command is executed

• Additionally, the drive can be configured to send automatically the error status, to a host. For that the "Send Data To Host" dialog (7) can be used.

4.3 Application evaluation, using the over-voltage protection

After the application runs, the drive will wait for the master to send the start command. To simulate the master, the "Command Interpreter" window can be used, to set the "cmd" variable to a value different than 0.

The motor will start to perform the motion profiles inside the "MachineProgram" loop. This can be checked, using the "1_Motion Status" control panel.

Command Interpreter		다. 말: 1_Motion St	tatus		8
TML> cmd = 1; TML>		15		15 Deference	Motor Current
	Ш	Position		Position	0.12
	Ш	2.493		2.468	
	l	,		,	[A]
	Ш	-15		-15	-10
	Ш	3000		3000	Speed Cmd
	Ш	Speed		Speed	100
	Ш	[rpm]		[rpm]	3000
	Ш	-90	· · · · · · · · · · · · · · · · · · ·	-100	- [
	l				_ [<mark>[rpm]</mark>
< >		-3000		-3000	-3000

Figure 21. Start the motion when "cmd" become different than 0

The over-voltage protection is continuously monitoring the motor supply voltage (AD4). When this value goes over the set threshold (UMAXPORT), the protection triggers and the drive executes the TML code inside the "Int 2. Software protections" interrupt routine.

To simulate an over-voltage situation, "UMAXPROT" should be set lower than AD4.



Figure 22. Trigger the over-voltage protection

The bit 12 in the MER error register (over-voltage) will be triggered and the drive will execute the code in the software protections interrupt.

SRH - Status Register High SRL - Status Register Low MER - Error Register DER - Detail Error Register DER2 - Detail Error Reg. 2 15 - Fault 0 15 - Axis is ON 0 15 - Enable is inactive 0 15 - EEPROM Locked 15 - Reserved 0 14 - In Cam 14 - Event set has occured 1 14 - Command error 0 14 - ENA hardware error 0 14 - Reserved 0 13 - In freeze control 0 18 - Moning/CALLS active 0 12 - Over voltage 1 13 - Self check error 13 - Reserved 0 11 - 12t warning - Drive 0 7 - Homing/CALLS warning 0 11 - Over temp Drive 0 11 - Start mode failed 11 - Reserved 0 10 - 12t warning - Motor 0 1 - Yes / True ENDINT 9 - UPD ignored for S-curve 9 - Reserved 0
15 - Fault 0 14 - In Cam 0 14 - In Cam 0 13 - In freeze control 0 14 - In Gear 0 12 - In Gear 0 11 - I2t warning - Drive 0 10 - I2t warning - Motor 0 9 - Target reached 0 10 - Not, False ENDINT 15 - Fault 15 - Enable is inactive 0 14 - Command error 0 14 - Cover voltage 1 11 - Over temp Drive 0 10 - Not or 1 10 - Not, False ENDINT 10 - New Strue ENDINT 10 - Net remp Motor 0 10 - Net remp Motor 0 10 - Served 0 10 - Net remp Motor 0 10 - Net remp.
14 - In Cam 0 13 - In freeze control 0 13 - In freeze control 0 12 - In Gear 0 11 - I2t warning - Drive 0 10 - I2t warning - Motor 0 9 - Target reached 0 10 - Notion is completed 0 12 - In Gear 0 13 - In Gear 10 - Notion is completed 0 11 - Over voltage 11 11 - Over temp Drive 0 10 - I2t warning - Motor 0 9 - Target reached 0
13 - In freeze control 0 12 - In Gear 0 12 - In Gear 0 11 - I2t warning - Drive 0 10 - Notion is completed 0 11 - I2t warning - Motor 0 9 - Target reached 0 10 - Notion is completed 0 11 - Ver voltage 1 12 - In Gear 0 13 - In Gear 0 14 - Ver voltage 1 15 - Ver voltage 1 16 - Ver temp Drive 0 17 - Homing/CALLS warning 0 11 - Over temp Drive 0 10 - Over temp Motor 0 10 - Ver temp Motor 0 10 - Seerved 0 10 - Ver temp Motor 0 9 - I2t 0 9 - UPD ignored for S-curve 0
12 - In Gear 0 8 - Homing/CALLS active 0 11 - I2t warning - Drive 0 10 - I2t warning - Motor 0 9 - Target reached 0 10 - I2t warning - Motor 0 9 - Target reached 0 10 - I2t warning - Motor 0 10 - Ver temp Motor 0 9 - Target reached 0 9 - Target reached 0 9 - Target reached 0 9 - Target reached 0 9 - Target reached 0 9 - Target reached 0 9 - Target reached 0 9 - Target reached 0 9 - Target reached <td< th=""></td<>
11 - 12t warning - Drive 0 7 - Homing/CALLS warning 0 10 - 12t warning - Motor 0 9 - Target reached 0 10 - 12t warning - Motor 0 11 - Start warning 10 - Ver temp Drive 10 - 12t warning - Motor 0 9 - Target reached 0 10 - Ver temp Motor 0 9 - Izt 0 10 - Ver temp Motor 0 9 - UPD ignored for S-curve 0 9 - Reserved 0
10 - 12t warning - Motor 0 Registers legend: 1 10 - Over temp Motor 0 10 - Encoder broken wire 0 10 - Reserved 0 9 - Target reached 0 9 - I2t 0 9 - UPD ignored for S-curve 0 9 - Reserved 0
9 - Target reached 0 1 - Yes / True ENDINIT 9 - 12t 0 9 - UPD ignored for S-curve 0 9 - Reserved 0
8 - Capture event/interrupt 0 8 - Nover current 0 8 - Invalid S-curve profile 0 8 - Reserved 0
7 - LSN event/interrupt 0 IMPORTANTI Check SRI.0 ! 7 - LSN (limit -) active 0 7 - Software LSN active 0 7 - Reserved 0
6 - LSP event/interrupt 0 supprivolage and some status of error bits are set ONLY after 6 - LSP (limit +) active 0 6 - Software LSP active 0 6 - Reserved 0
5 - Autorun enabled 1 ENDINT is executed. 5 - Feedback error 0 5 - Cancelable call ignored 0 5 - Hall sensor missing 0
4 - Over position trigger 4 1 BasySetUp, download a setup, EasySetUp, download a setup, 2 - Serial comm. error 0 4 - UPD ignored 0 4 - AEI interface error 0
3 - Over position trigger 3 1 reset the drive and press the arrow 3 - Control error 0 3 - Function not available 0 3 - BiSS sensor missing 0
2 - Over position trigger 2 1 nearby button to send an ENDINI 2 - Invalid setup data 0 2 - Homing not available 0 2 - BISS data error 0
1 - Over position trigger 1 1 EasyMotion Studio, run a TML 1 - Short-circuit 0 1 - TML stack underflow 0 1 - BISS data warning 0
0 - ENDINI executed 1 program. This includes execution 0 - CANbus error 0 0 - TML stack overflow 0 0 - BISS CRC error 0
Supply voltage 24.007 [V] Drive temperature 43.238 [°C]

Figure 23. Over-voltage protection

Setting the "UMAXPROT" parameter value to the default value, the over-voltage state will disappear and the drive will execute the error recovery procedure.

Command Interpreter	- • •
TML> cmd = 1;	~
IML> 2AD4 AD4 (uint@0x0240) = 15360 (0x3C00)	
TML> ?UMAXPROT	
UMAXPROT (uint@0x029A) = 33945 (0x8499)	
TML> UMAXPROT = 1000 TML> UMAXPROT = 33945	
TML>	
<	>

Figure 24. Reset UMAXPROT to the initial value

4_Drive Status					×
SRH - Status Register High	SRL - Status Register Low MER - Error Register		DER - Detail Error Register	DER2 - Detail Error Reg	j. 2
15 - Fault	15 - Axis is ON 1	15 - Enable is inactive	15 - EEPROM Locked 0	15 - Reserved	0
14 - In Cam 0	14 - Event set has occured 1	14 - Command error 0	14 - ENA hardware error 0	14 - Reserved	0
13 - In freeze control	10 - Motion is completed 1	13 - Under voltage 0	13 - Self check error 0	13 - Reserved	0
12 - In Gear	8 - Homing/CALLS active	12 - Over voltage	12 - TML heartbeat ignored 0	12 - Reserved	0
11 - I2t warning - Drive	7 - Homing/CALLS warning 0	11 - Over temp Drive 0	11 - Start mode failed 0	11 - Reserved	0
10 - I2t warning - Motor	Registers legend:	10 - Over temp Motor 0	10 - Encoder broken wire 0	10 - Reserved	0
9 - Target reached 1	1 - Yes / True ENDINIT	9 - 12t 0	9 - UPD ignored for S-curve 0	9 - Reserved	0
8 - Capture event/interrupt 0		8 - Over current 0	8 - Invalid S-curve profile 0	8 - Reserved	0
7 - LSN event/interrupt	IMPORTANT! Check SRH.0 !	7 - LSN (limit -) active 0	7 - Software LSN active 0	7 - Reserved	0
6 - LSP event/interrupt 0	or error bits are set ONLY after	6 - LSP (limit +) active 0	6 - Software LSP active 0	6 - Reserved	0
5 - Autorun enabled 1	ENDINIT is executed.	5 - Feedback error 0	5 - Cancelable call ignored 0	5 - Hall sensor missing	0
4 - Over position trigger 4 1	If SRH.0 = 0 and you use EasySetUp, download a setup.	4 - Serial comm. error 0	4 - UPD ignored 0	4 - AEI interface error	0
3 - Over position trigger 3 1	reset the drive and press the	3 - Control error 0	3 - Function not available 0	3 - BiSS sensor missing	0
2 - Over position trigger 2 1	nearby button to send an ENDINIT	2 - Invalid setup data 0	2 - Homing not available 0	2 - BiSS data error	0
1 - Over position trigger 1 1	EasyMotion Studio, run a TML	1 - Short-circuit	1 - TML stack underflow	1 - BiSS data warning	0
0 - ENDINIT executed 1	program. This includes execution of ENDINIT.	0 - CANbus error 0	0 - TML stack overflow 0	0 - BiSS CRC error	0
Supply voltage 23.932 [V]	Drive temperature 47.429 [°C]				

Figure 25. The error state resetting

Once the error recovery procedure is executed and the fault state is reset, the drive will return to the "MasterCommand" loop (in the main program) and wait again for the master command.

5. Conclusions

In case of the complex systems, controlled by a PC / PLC master, is recommended to distributed the intelligence between the master and the slave devices.

In this application, the master sends only the start command and needs to take some decisions when a fault occurs. The rest of the tasks were implemented on the Technosoft drive.

This solution minimizes the communication channel traffic and the tasks that needs to be executed by the master.

In case of the masters where the TML protocol needs to be implemented, this structure has also the advantage that requires to implement (at the master level) only a minimum set of TML instructions.