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

![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](image)

Figure 3. Main section of the TML program
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 Dialogue](image)

![Assignment & Data Transfer - 16 bit Integer Data Dialogue](image)

- 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.

![Figure 5. How to define variables](image)

![Figure 6. How to set a value for a variable](image)
• 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](image)

**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 Profiles](image)

**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.

![Figure 9. How to create an event](image)

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.

**Remark:** While an interrupt is active, the other interrupts are deactivated. It is recommended to keep the ISR as short as possible. If it is not possible, then the other interrupts should be re-enabled using the “Interrupts Settings” dialogue.

![Figure 10. Interrupt Settings dialog](image)

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.

![Figure 11. How to set output line status](image)

• 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.

![Figure 12. How to deactivate the drive power stage](image)
• 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.

![Figure 13. How to set a value for a variable](image1)

• 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.

![Figure 14. How to implement a TML loop](image2)
• 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 & Data Transfer - 16-bit Integer Data dialogue](image)

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

**Remark:** 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 is presented in the EasyMotion Studio help topics [link].

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

![Events dialogue](image)

**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.

![Motion - Trapezoidal Profiles dialogue](image)

**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](image)

**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.

![I/O](image)

**Figure 19. How to reset the output line status**

The fault status is reset using the “FAULTR;” TML instruction (see the “Miscellaneous” (8) dialog).
Remark: 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.

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.
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.

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