TECHNOSOFT
MOTION TECHNOLOGY

Drive special inputs
Enable

Application Note

Easy Motion Studio II

TECHNOSOFT
MOTION TECHNOLOGY

Table of content

1. Introduction

The Enable input allows connecting an emergency signal that can come from the master or simply from an emergency button.

When the Enable input becomes “inactive”, the drive performs the following actions:

- deactivates the PWM outputs
- sets the bit 15 (“Enable is inactive”) in the MER (Motion Error Register) error register
- generates the “int0 – Enable input has changed” TML interrupt

Remark: With the PWM outputs being deactivated, the motor will not be energized anymore, so it will stop freely, becoming fully dependent on the load inertia and system friction.

When the Enable input changes back to “active”, the MER.15 bit is reset but the PWM outputs remain deactivated, with one exception: when the drive was executing an electronic gearing profile.

The “int0 – Enable input has changed” interrupt is triggered when the Enable input becomes active.

This application note presents an example on how to restore a predefined motion cycle (stored at the drive level), when the Enable input is reactivated.

2. Application flow chart

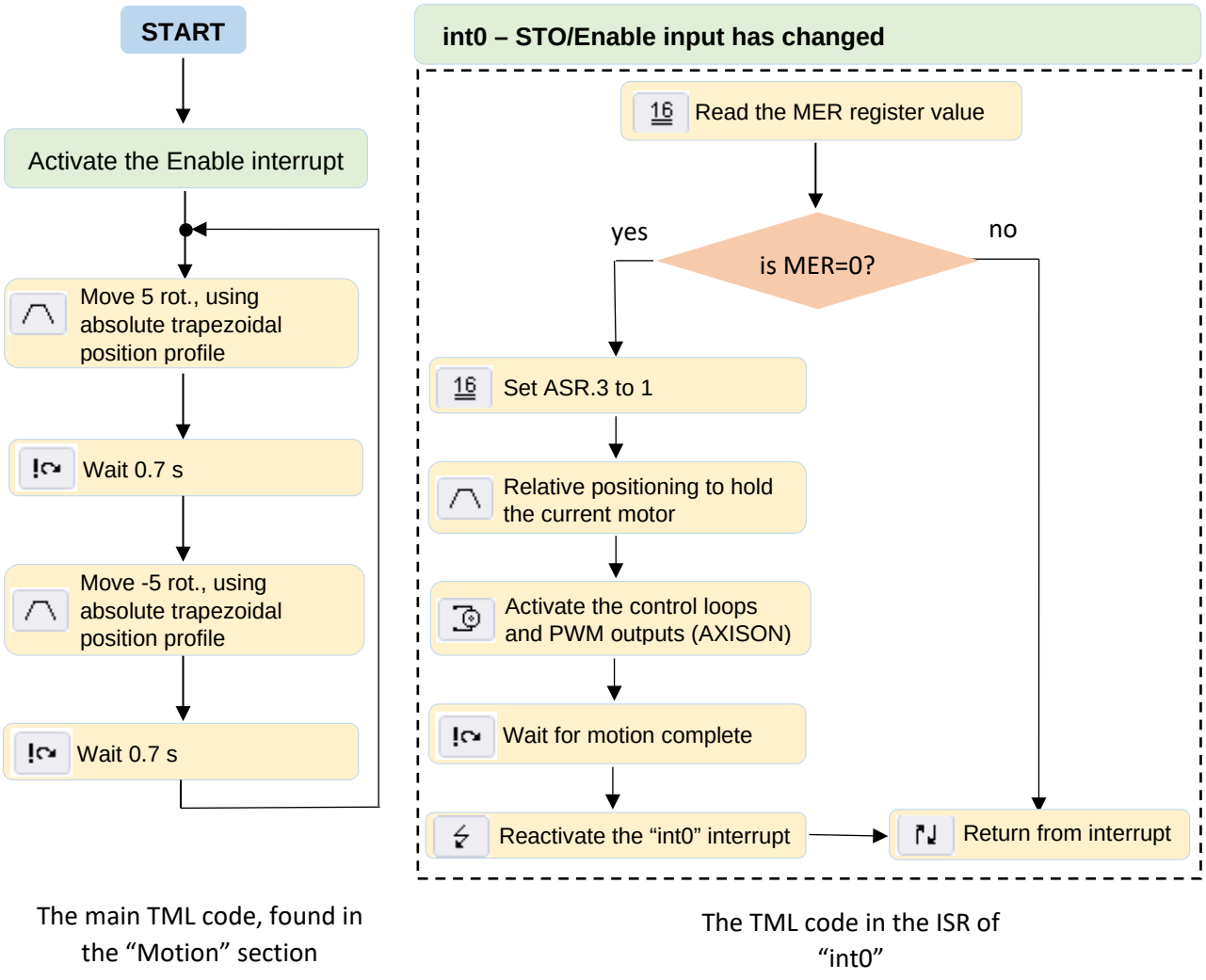

Figure 1 - Application flow chart

3. EasyMotion Studio II implementation

The application implementation will follow the application flow chart from section 2 and has two parts:

1) The main TML program, in the “Motion” branch, that will include the predefined motion cycle.
2) The customized ISR for “int0 – Enable input has changed”.

3.1 Main motion program

The main program contains an infinite loop with two absolute motion profiles. The first profile will move the load to 5 rotations, while the second one will move it back to the start position. Between the motion profiles, a 0.7 s waiting time will be introduced.

Remark: The TML user variables must be defined at the beginning of the “Motion” section including the ones used in custom ISR or TML functions.

The code sequence inside the “Motion” section was generated using the buttons marked with 1 to 8 in Figure 9. Clicking on those buttons the following programming dialogs will open:

a) The user variables can be declared with the “Miscellaneous” dialogue (Figure 9 – step 1). Here, it was used to declare the “Mer_copy” variable required in the “int0 – Enable input has changed” interrupt service routine. Mer_copy will hold the value of MER register, integer 16-bit, therefore it will be defined as integer too.

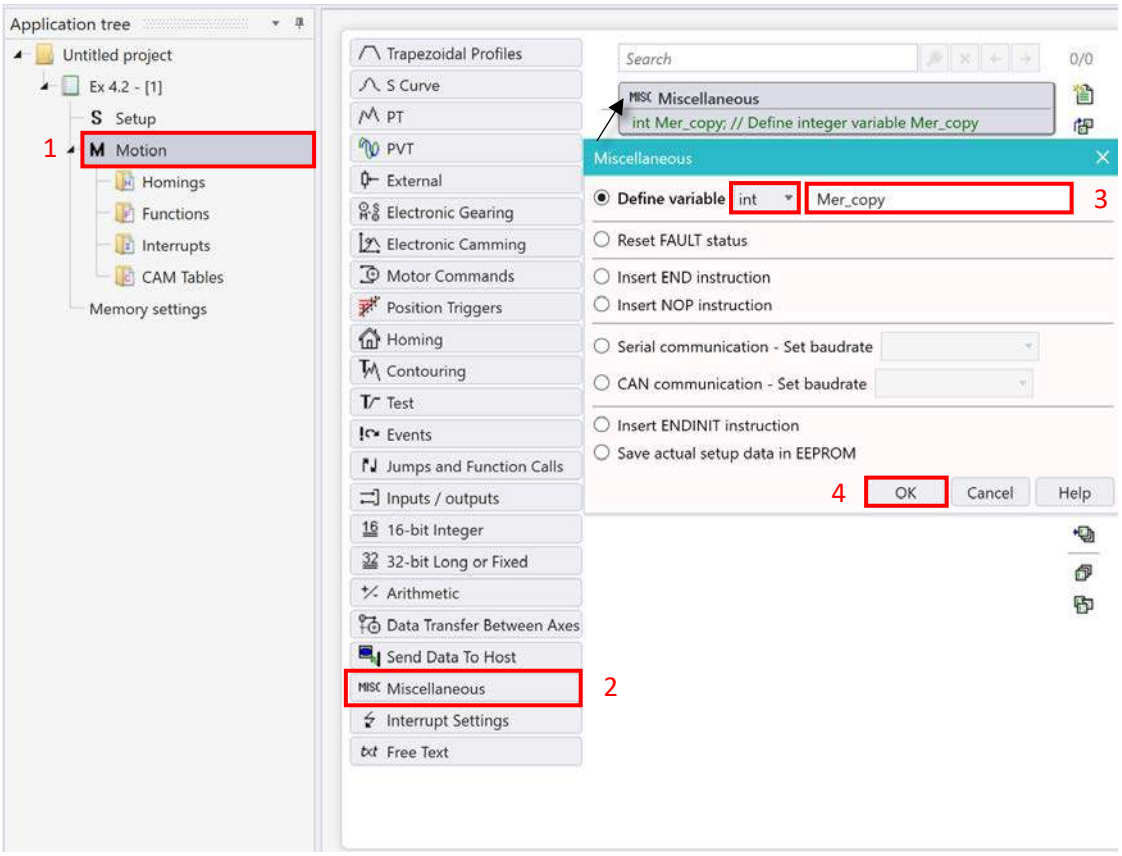

Figure 2 - How to define a user variable, step by step. (Figure 9 – step 1)

b) The “Interrupt Settings” dialog (Figure 9 – step 2) allows to enable, disable and clear previous requests of the TML interrupts. In this case it was used to activate the “Int0 – Enable input has changed” interrupt routine, that was used to implement the functionality described in the first section.

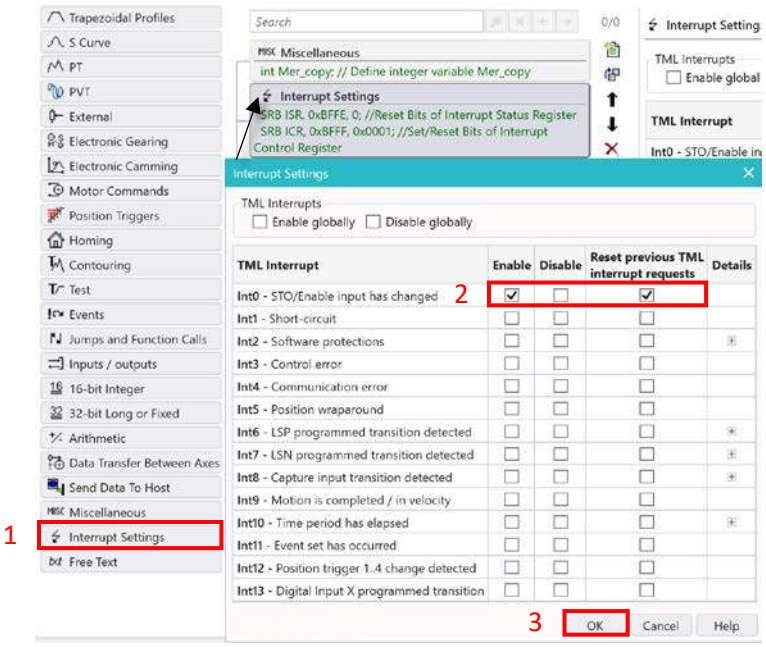

Figure 3 - How to activate the enable input interrupt (int0). (Figure 9 – step 2)

c) The “Jumps and Function Calls” dialogue (Figure 9 – step 3 and step 8) allows controlling the TML program flow through unconditional or conditional jumps and unconditional, conditional or cancelable calls of TML functions.
In this case, the dialog was used to create an infinite loop where the motor executes 5 rotations in the positive direction and returns to the start position. Therefore, the remaining lines of code will be placed between the codes illustrated in Figure 4 and Figure 5. In other words, we will start with “Loop_01”, write the remaining lines of code, and end with “GOTO Loop_01”.

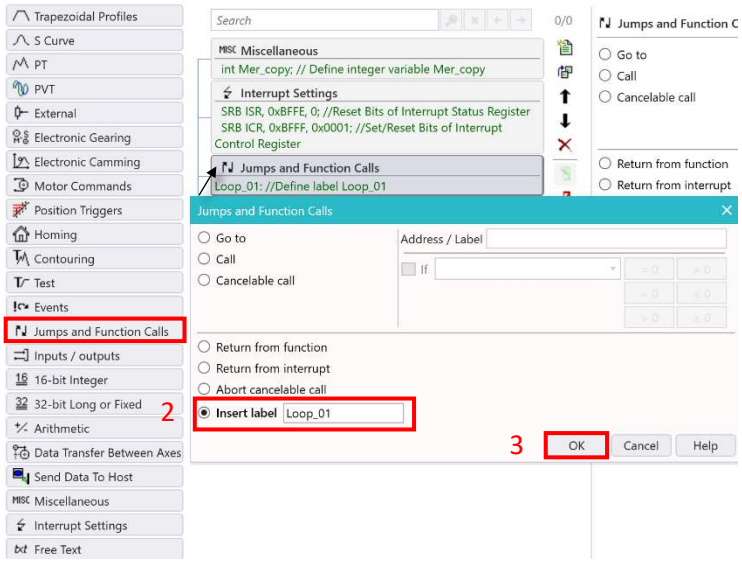

Figure 4 - Inserting the label for a TML loop. (Figure 9 – step 3)

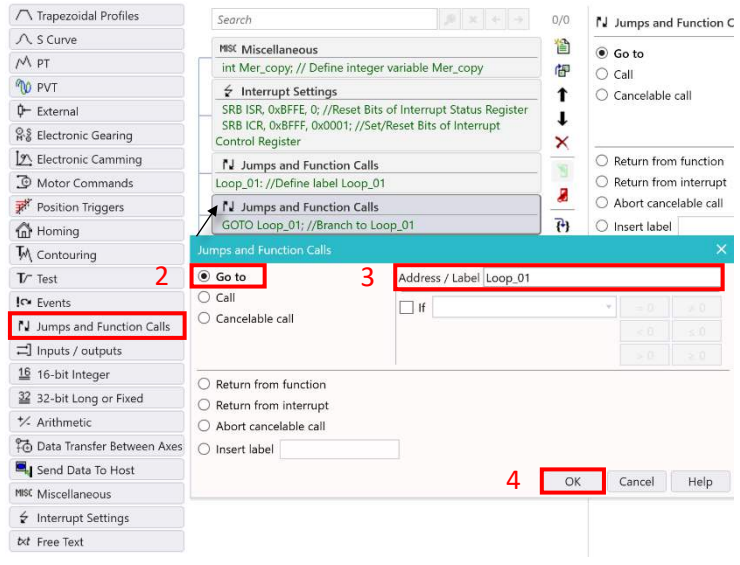

Figure 5 -Inserting the command to go back to the label in a TML loop. (Figure 9 – step 8)

Remark: For visual esthetics and easiness of read reasons, all figures from this point on will not showcase anymore the previously generated lines of code, as they will be grouped (displayed with the blue “Untitled” text). To make sure the instructions have been placed in the right order, see Figure 9.

d) The “Motion – Trapezoidal Profiles” dialogue (Figure 9 – step 4 and step 6) allows to program a position or speed profile with a trapezoidal shape of the speed, due to a limited acceleration.
In this case it was used to move the motor for 5 rotations and then to return to 0.

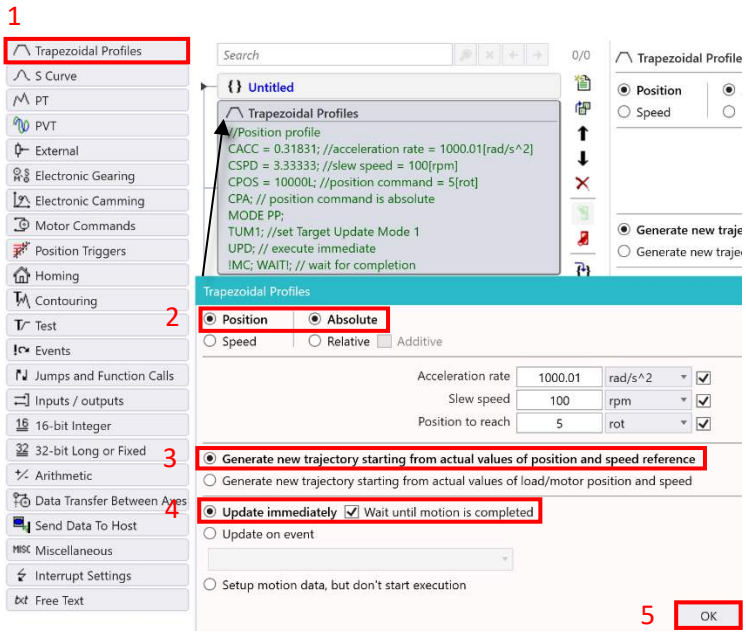

Figure 6 - Configuration for 5 absolute positive rotations of the rotor. (Figure 9 – step 4)

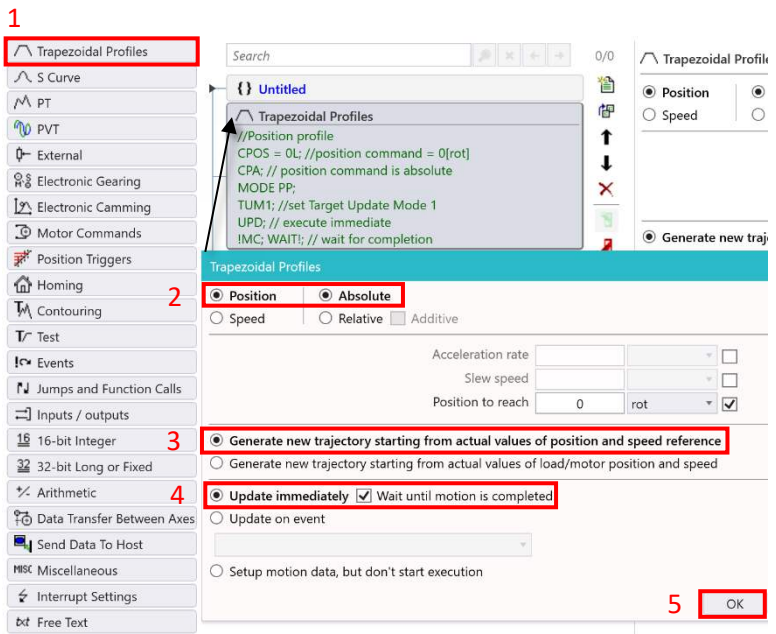

Figure 7 - Configuration for returning to the initial absolute position. (Figure 9 – step 6)

e) The “Events” dialogue (Figure 9 – step 5 and step 7) allows defining events. An event is a programmable condition, which once set, is monitored for occurrence.

The following actions can be executed automatically when an event is detected:
- stop the motion when the event occurs.
- wait for the programmed event to occur.

In this case, the “Events” dialog was used to insert a 0.7 s delay between the two movements.

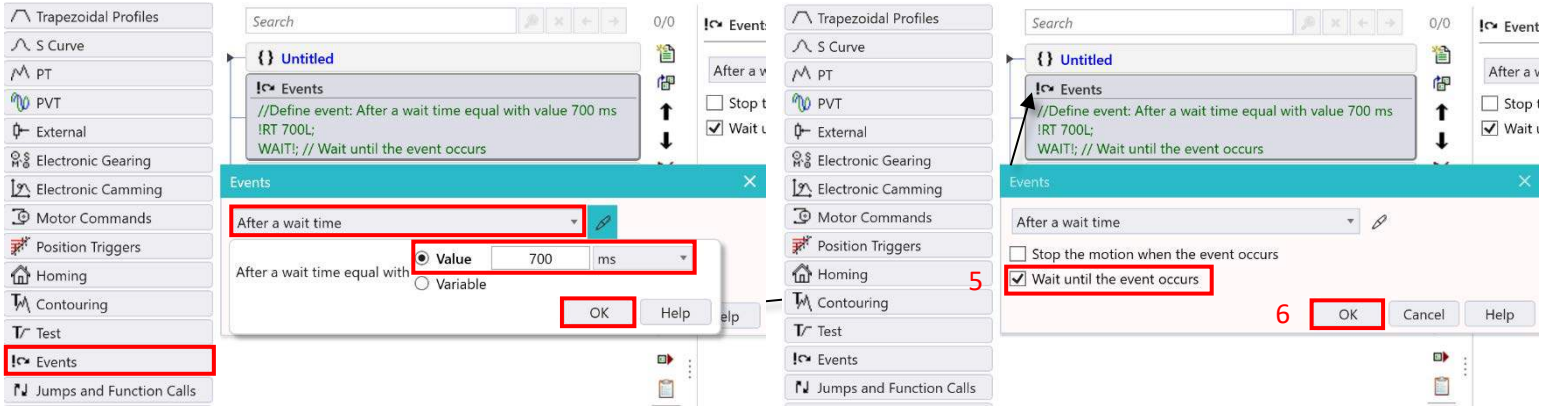

Figure 8 – How to insert a delay in TML program. (Figure 9– step 5 and step 7)

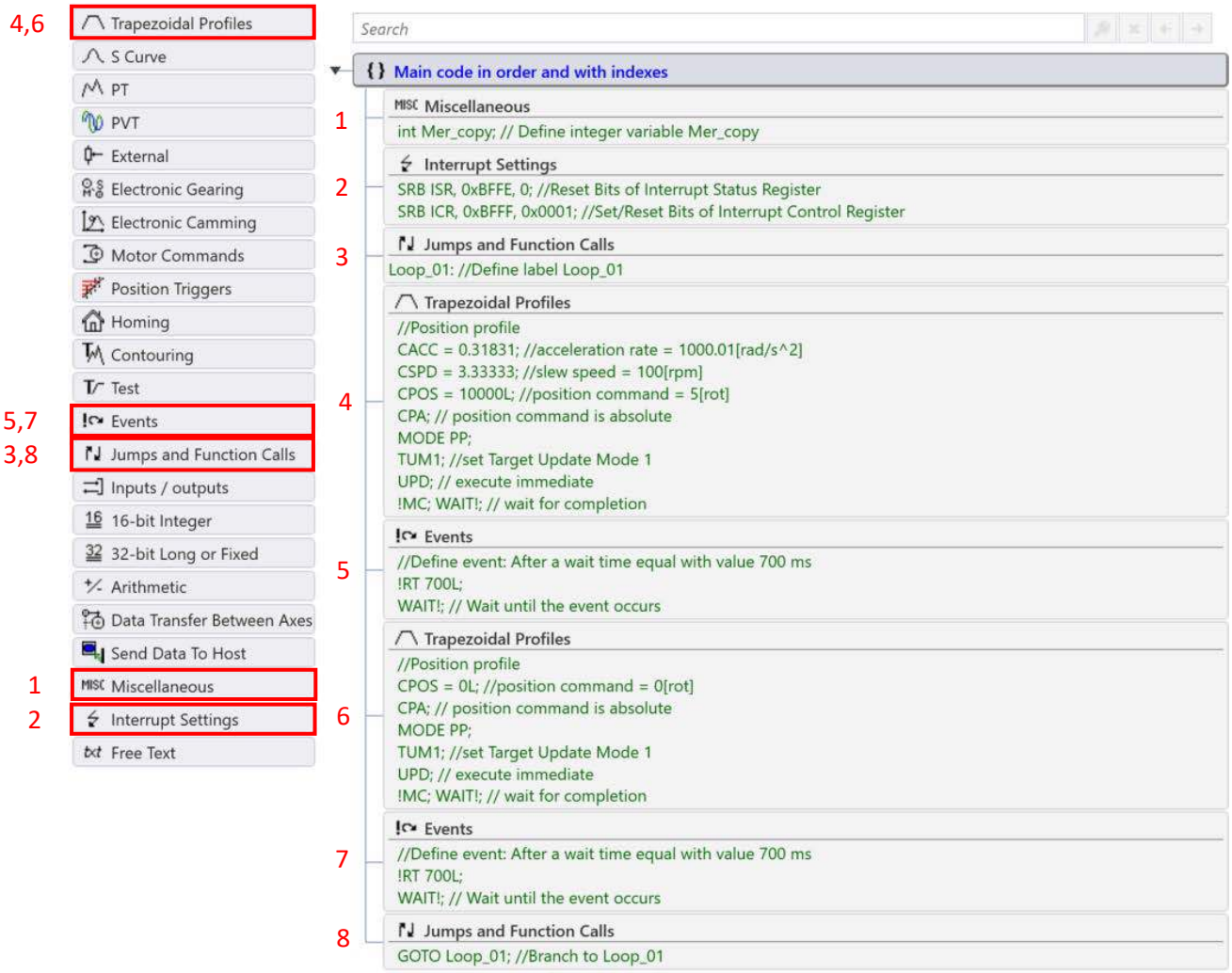

Figure 9 – Complete TML code with indexes.

3.2 Customized interrupt service routine the “int0 – Enable input has changed” TML interrupt

The TML interrupts are special functions executed automatically when certain conditions are detected by the drive. 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: During the execution of an ISR, the other interrupts are deactivated. Hence, 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.

This application was implemented using the “Int0 – Enable input has changed” interrupt.

The TML code from the “Int0 – Enable input has changed” ISR was generated using the buttons marked with 1 to 6 in Figure 3. Clicking on those buttons the following programming dialogs will open.

a) The “Assignment and Data Transfer – 16-bit Integer Data” dialogue (Figure 20 – step 1 and step 2) allows various manipulations of 16-bit integer variable/parameter/register.

   Here, the “Assignment and Data Transfer – 16-bit Integer Data” dialogue was used to copy the value of the MER error register in “Mer_copy” isolate bit 15 (“Enable is inactive”) with “AND” and “OR” masks.

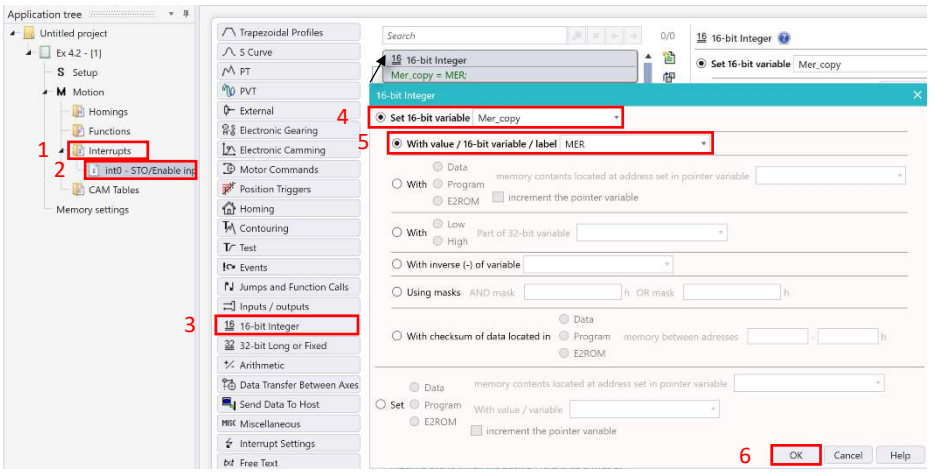

Figure 10 - Creating the copy of the internal "MER" variable with the user defined variable "Mer_copy" – (Figure 20 – step 1)

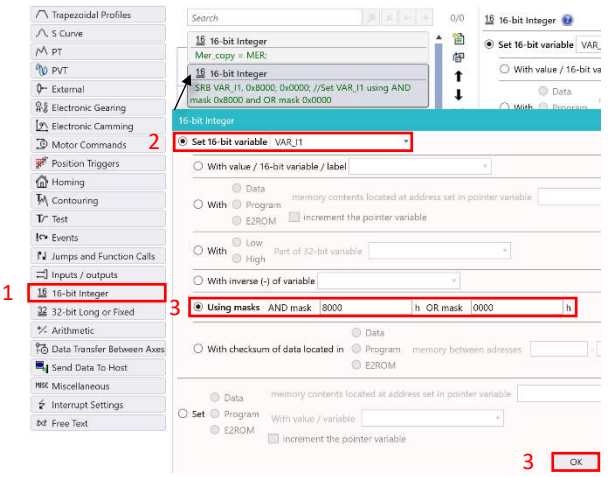

Figure 11 - Adding a mask to the assigned value for variable "Mer_copy" so that we only save the value of bit 15. (Figure 20 – step 2)

The description of MER can be found with “Help | Help Topics | Application Programming | Technosoft Motion Language | Basic Concepts” as shown in Figure 11 and then select TML Registers under TML Data in help tree.

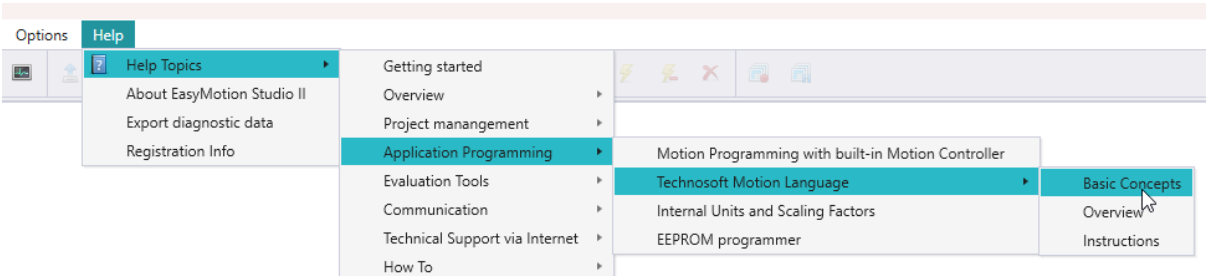

Figure 12 - Accessing the EasyMotion Studio II Help.

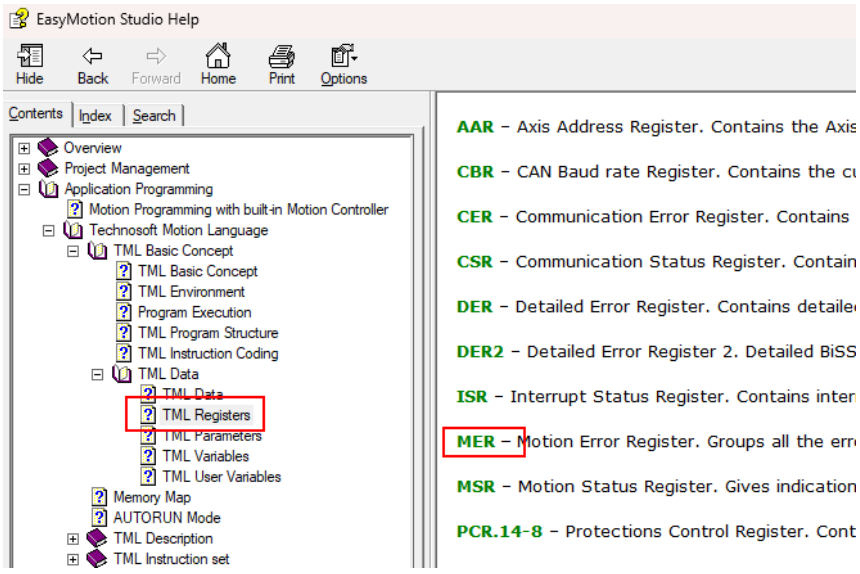

Figure 13 - TML registers topic inside built-in help

Contents: MER information is structured as follows:

15	14	13	12	11	10	9	8
ENST	CMDER	UVER	OVER	OTERD	OTERM	I2TER	OCER
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
LSNST	LSPST	FDBKER	SCIER	CTRER	STPTBL	SCER	CANBER
0	0	0	0	0	0	0	0

Bit 15 ENST. Enable status of drive/motor

0 = Enabled
1 = Disabled

From this point on we will present only the blocks used for each line of code, in the right order. To check again if you placed them correctly, check Figure.

b) The “Jumps and Function Calls” dialogue (Figure 20 – step 3 and step 4) was used to generate the “GOTO AXSIENABLE;” and “RETI;” instructions that makes the program to jump to the “AXISENABLE;” label, if MER.15 is 1 or to return from the interrupt (“RETI;”) if MER.15 is 0.

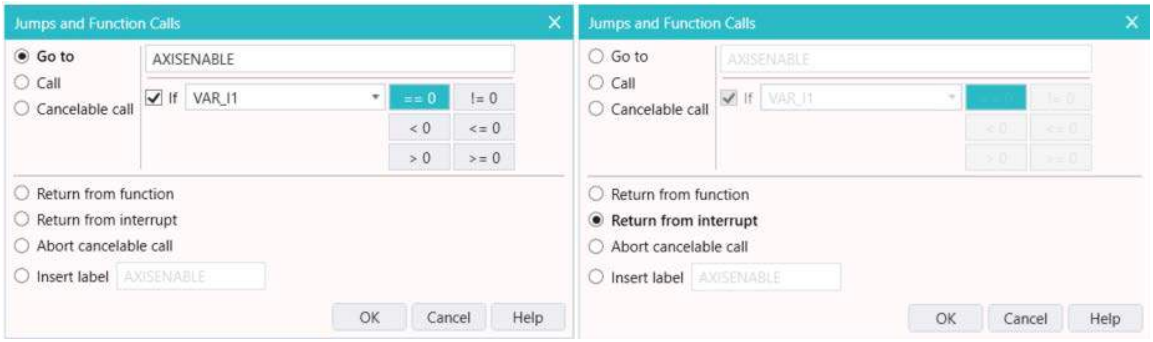

Figure 14 - Insert a "GOTO" (Figure 20 – step 3) and "RETI" (Figure 20 – step 4) TML instruction.

The “AXISENABLE” label was also created using the “Jumps and Function Calls” dialogue (Figure 20 – step 5).

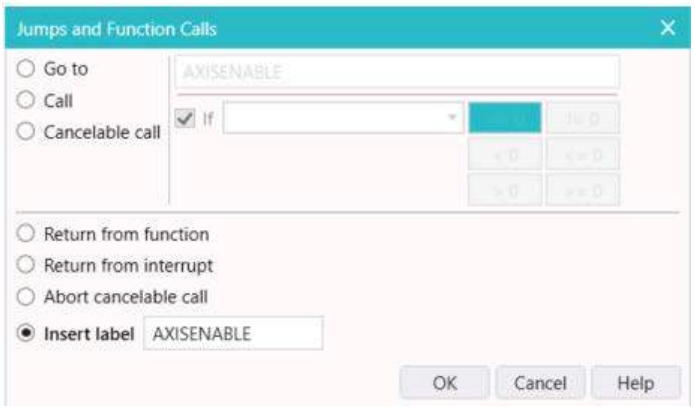

Figure 15 - How to create a label. (Figure 20 – step 5)

c) The “Assignment and Data Transfer – 16-bit Integer Data” dialogue (Figure 20 – step 6) was used again, to set the bit 3 from the ASR register with “AND” and “OR” mask. This will make the target speed 0, when the next position profile (with TUM0 - the reference is generated starting from the actual value of load/motor position and speed) is executed. The purpose of this instruction is to prevent the motor from moving to the last imposed position, when the drive power stage is reactivated.

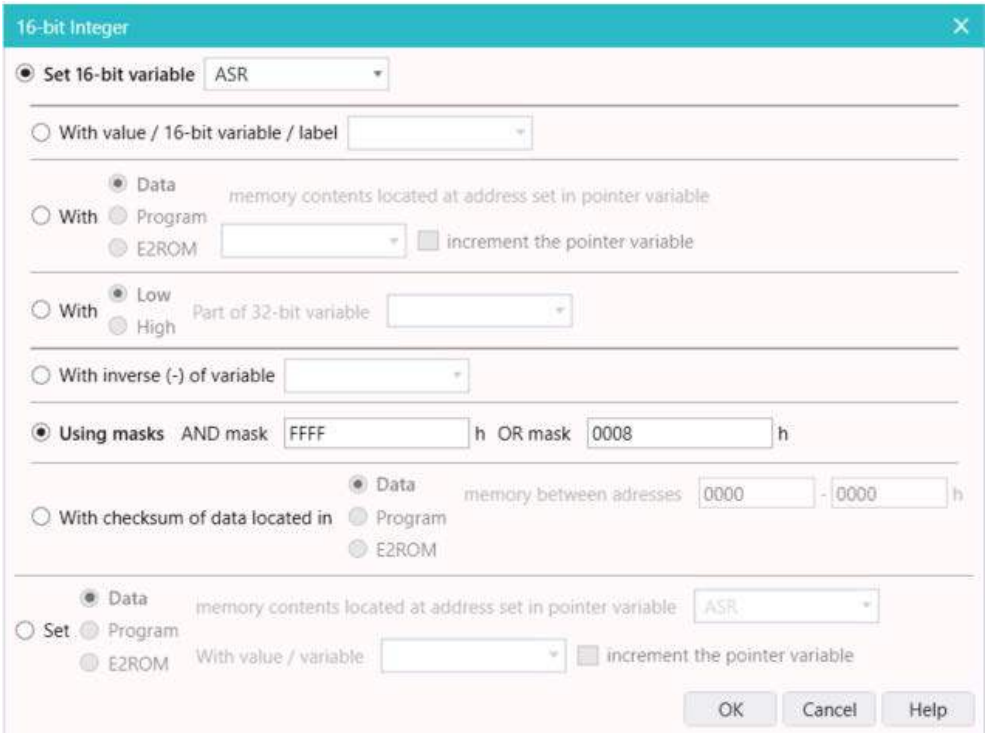

Figure 16 - Set the bit 3 in the ASR register to 1. (Figure 20 – step 6)

d) The “Motion – Trapezoidal Profiles” dialog (Figure 20 – step 7) was used to insert a relative position profile, with “TUM0” and a position increment of 0 rot. This way the motor will hold the current position once the axis is re-enabled.

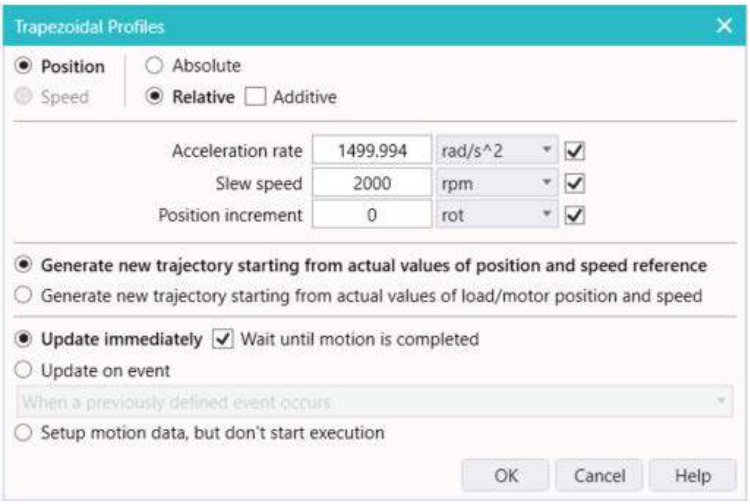

Figure 17 - How to configure and start motion using a position profile using TUM0.

e) The “Motion - Motor Commands” dialogue (Figure 20 – step 8) is used to reactivate the drive PWM outputs and allow the position profile above to start being executed.

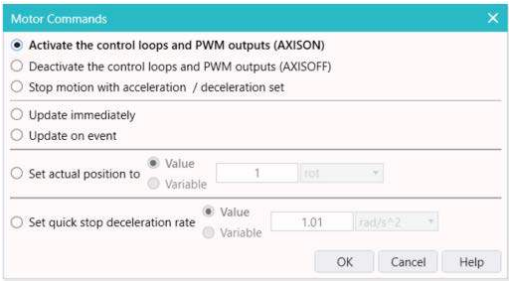

Figure 18 - How to configure and start motion using a position profile using TUM0. (Figure 20 – step 8)

f) The “Events” dialogue (3) was used here to hold the program execution until the previous motion profile (with CPOS = 0 rpm) is executed and the Motion Complete bit is set to 1.

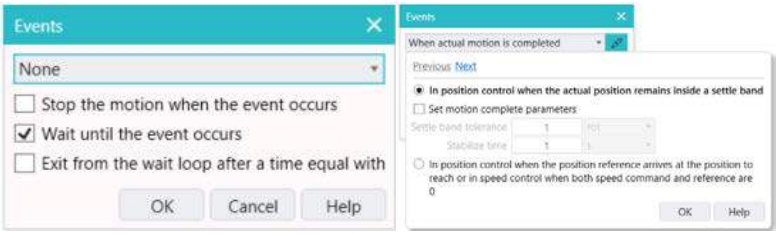

Figure 19 - Set an event on Motion Complete. (Figure 20 - step 9)

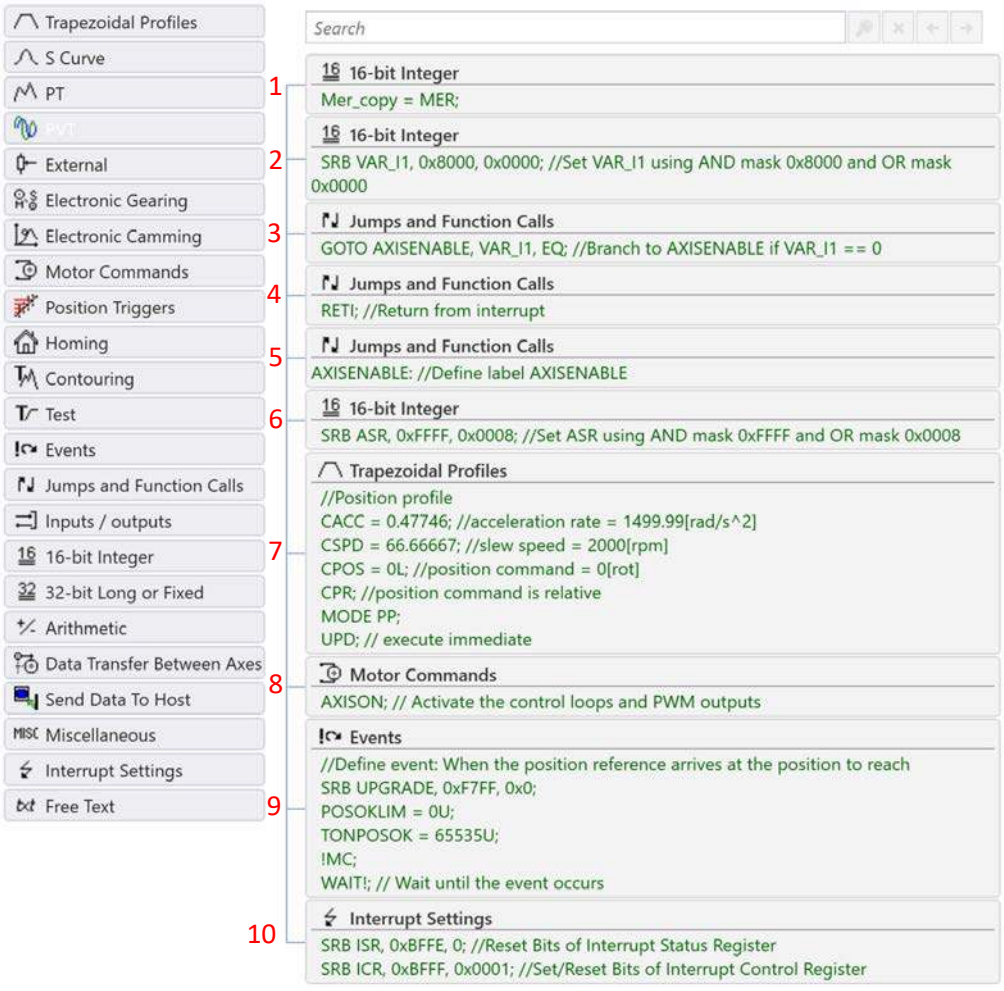

Figure 20 - Entire code for the custom "int0".

Once the code under the “AXISENABLE” label is executed, the drive power stage (the PWM outputs) will be reactivated, and the motor will hold the current position.

The program will return from the “Int0 – Enable input has changed” interrupt routine, to the last executed instruction in the main TML program. As most probably this instruction is one of the two trapezoidal position profiles in the “Loop_01” loop, the motor will start to spin again, performing the back and forward motion inside the loop.

Remark: In case the ENABLE input needs to be used as a general-purpose input, its default behavior, can be disabled by setting the “ENABLEOFF” parameter to 1.

TECHNOSOFT

MOTION TECHNOLOGY