1. Application description

This application note describes how to activate and customize the TML interrupt services routines, using an example that sets the “Int 10 – Time period has elapsed” interrupt, to flash a LED, connected to one of the drive digital outputs.

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.

While an interrupt is active, the other interrupts are deactivated. That is why, it is recommended to keep the ISR as short as possible. If this is not possible, then the other interrupts should be re-enabled using the “Interrupts Settings” dialogue (will be presented in chapter 4).

2. Application flow chart

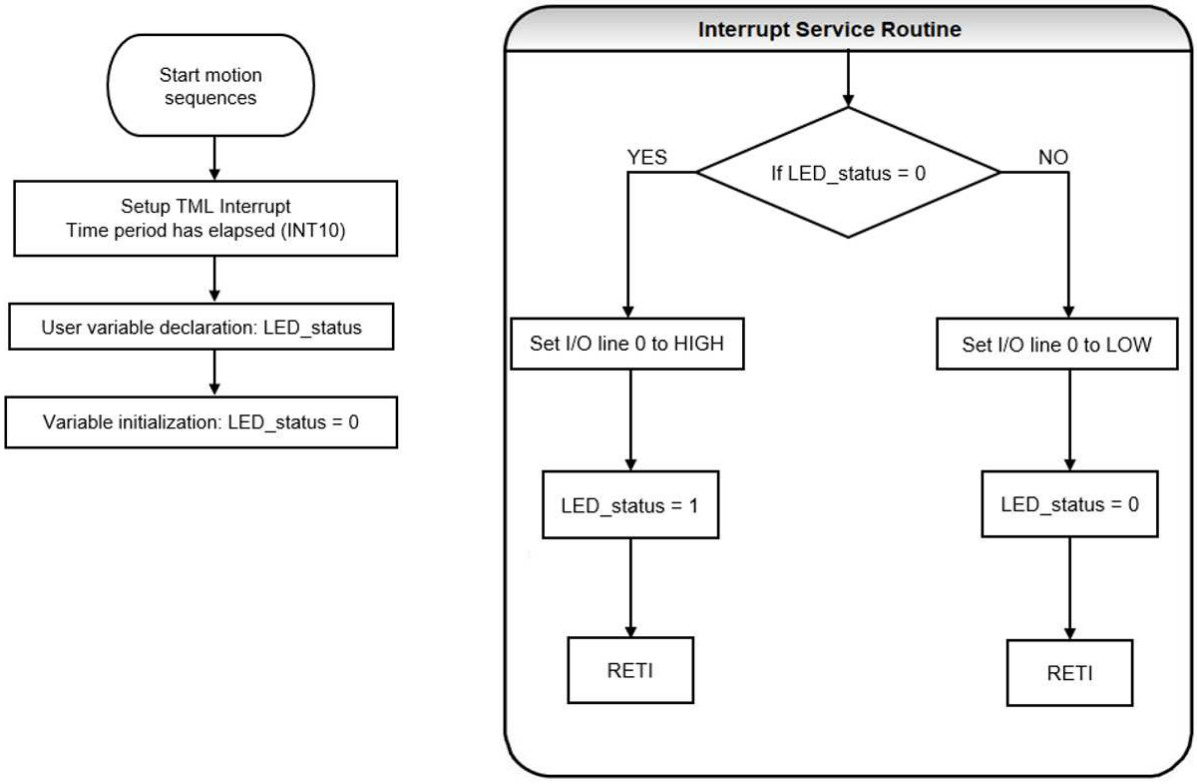

Figure 1. *Application structure*

3. EasyMotion Studio implementation

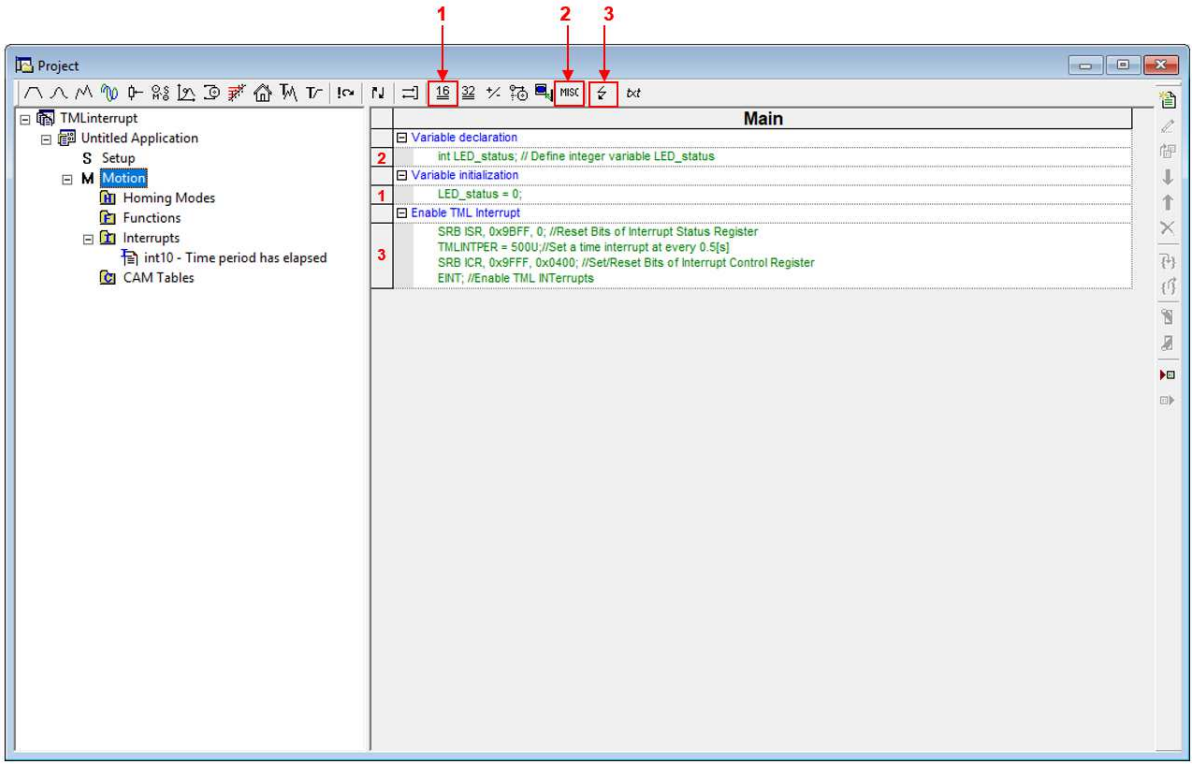

Figure 2. *Main section of the TML program*

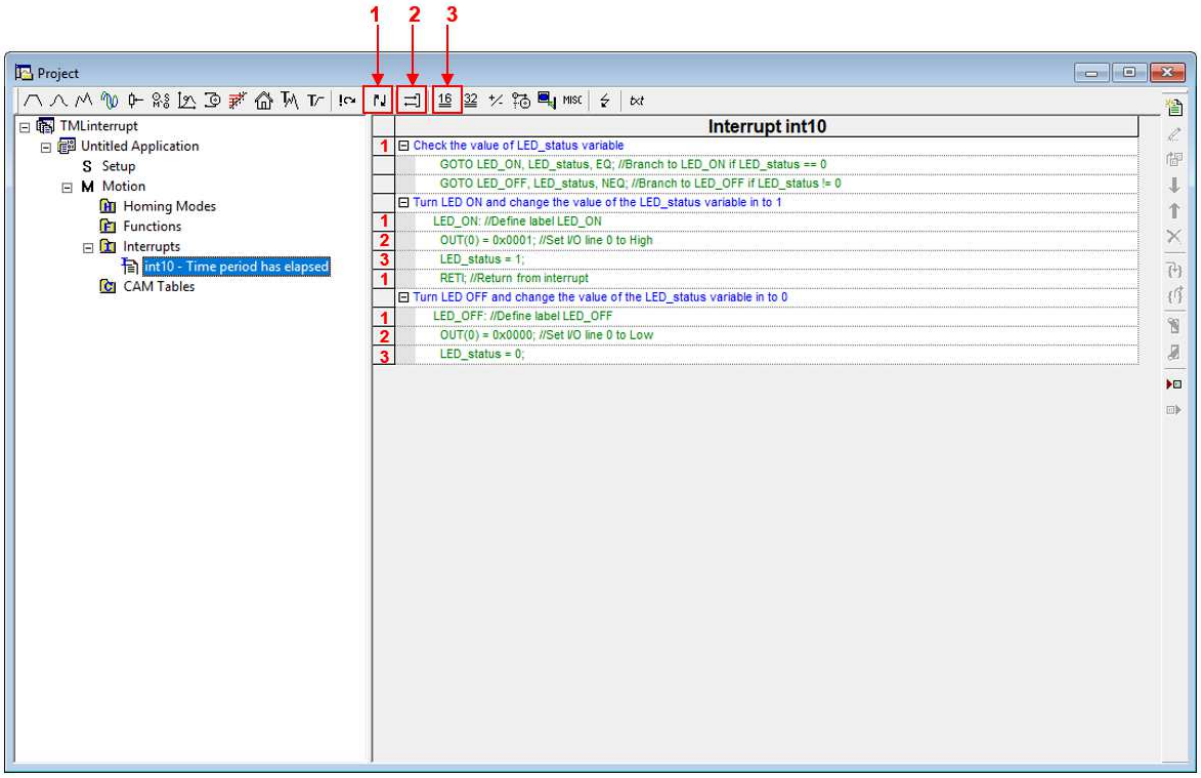

Figure 3. *Time interrupt service routine section*

4. Detailed description of the EasyMotion Studio implementation

4.1 Motion section

The code sequences in the “Motion” section were generated using the buttons marked with 1 to 3 in Figure 2. Clicking on those buttons the following programming dialogues will open.

• The “Miscellaneous” dialogue (2) allows to declare user variables, reset/exit the drive/motor from the fault status, execute the “END” / “NOP” / “ENDINIT” TML instructions, change the CAN / RS-232 baudrate and save the actual setup into the drive memory.

In this case the “Miscellaneous” dialogue was used to declare “LED_status” user variable, that is use to track the output status (active or inactive).

• The “Assignment and Data Transfer – 16 bit Integer Data” dialogue (1) allows different operations with the 16-bit integer variables / parameters / registers. Here it was sued to initialize the “LED_status” user variable with 0.

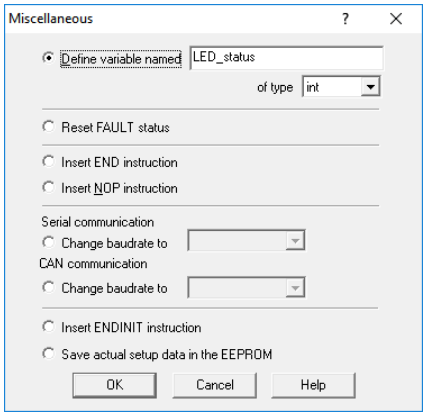

Figure 4. *Defining the user variables*

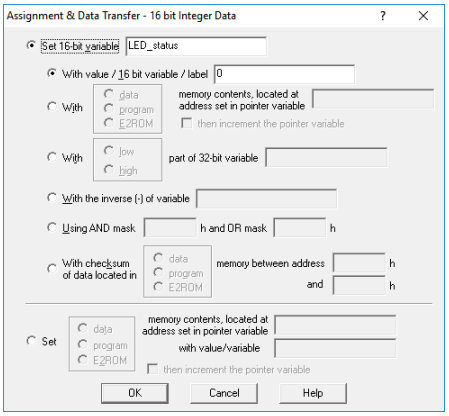

Figure 5. *Set the a user variable*

• The “Interrupt Settings” dialogue (3) allows to activate and/or deactivate the TML (Technosoft Motion Language) interrupts. In this case, it was used to activate the “int10 – Time period has elapsed” interrupt routine and set it to 0.5 s.

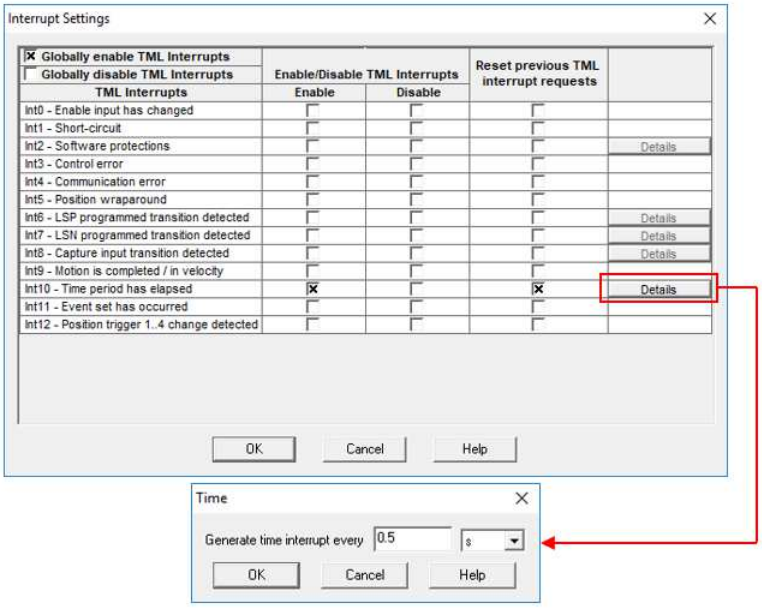

Figure 6. *Interrupt Settings dialogue*

4.2. Time period Interrupt routine

The “Interrupts” section allows to customize the TML interrupt service routines. Once the "User defined" option is marked the interrupt routine will appear in the project window (left side), under the “Interrupts” section.

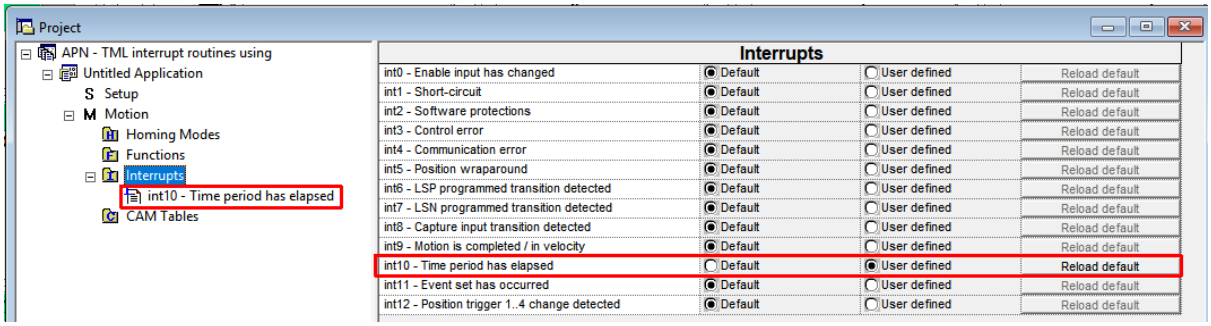

Figure 7. *How to customize the TML interrupt service routines*

This application uses the “Int10 – Time period has elapsed” interrupt routine, to check the value of the “LED_status” user variable and command the OUT(0) digital output.

The code sequence inside the “Int10 – Time period has elapsed” interrupt was generated using the buttons marked with 1 to 3 in Figure 3. Clicking on those buttons the following programming dialogues will open.

• The “Jumps and Function Calls” dialogue (1) allows to control the TML program flow through unconditional or conditional jumps and unconditional, conditional or cancelable calls of TML functions. In this application, the “Jumps and Function Calls” dialogue was used to create some conditional jumps, function on the “LED_status” user variable (if “LED_status = 0” the “OUT(0)” output is set to the active level, to switch ON the LED. Otherwise, the “OUT(0)” is set inactive, to switch OFF the LED).

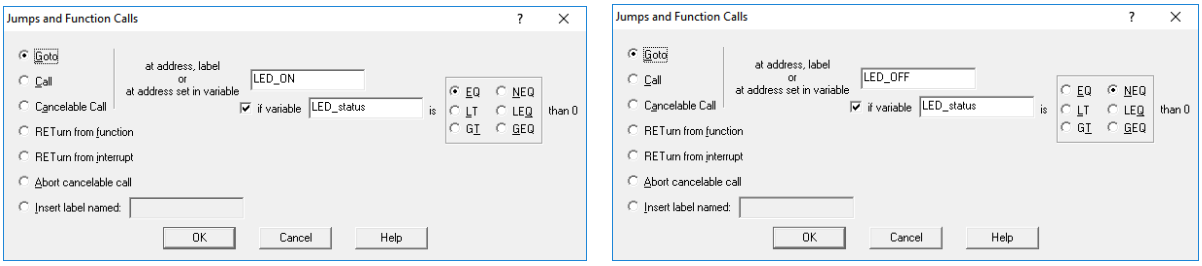

Figure 8. *Jump to “LED_ON” / “LED_OFF” label, function on the “LED_status” user variable*

The same dialogue was used to create the “LED_ON” and “LED_OFF” labels.

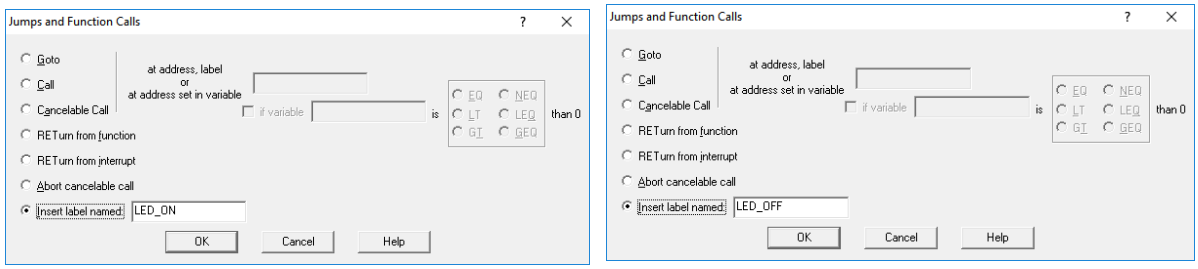

Figure 9. *Create the “LED_ON” / “LED_OFF” label*

• The “I/O” dialogue allows different operations with the drive digital inputs and outputs. It was used here to set the “OUT(0)” digital output LOW or HIGH (function on the “LED_status” variable value). This way, the LED connected to this input is switched ON or OFF.

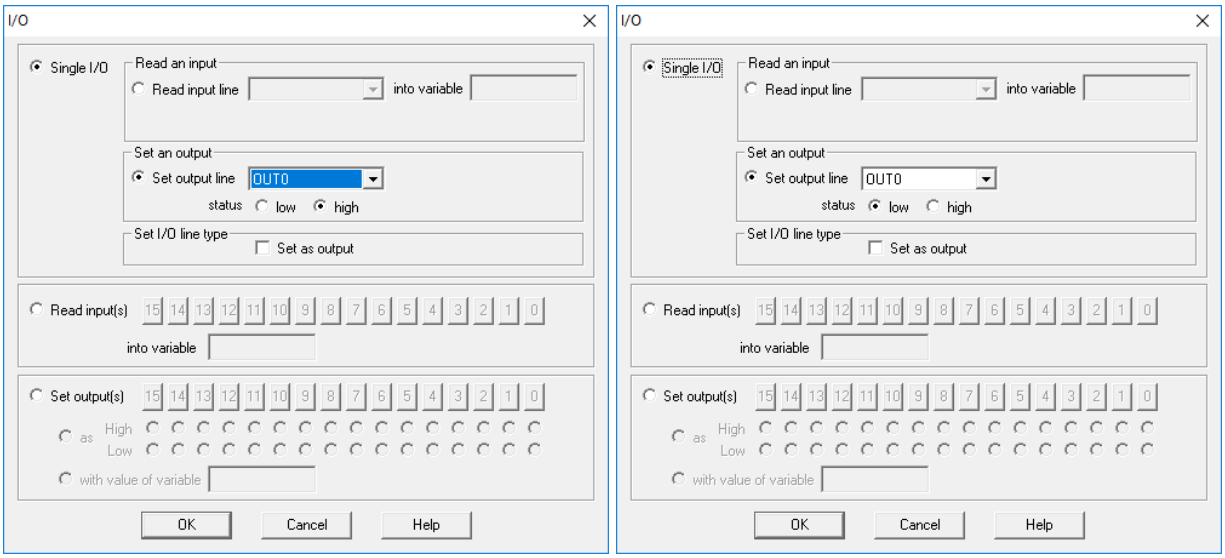

Figure 10. *How to set output line status*

• Once the OUT(0) digital output status is changed, the “Assignment and Data Transfer – 16 bit Integer Data” dialogue (3) is used to modify the “LED_ststus” variable value. This has the purpose to indicate that the LED is ON (“LED_status = 1”) or OFF (”LED_status = 0”).

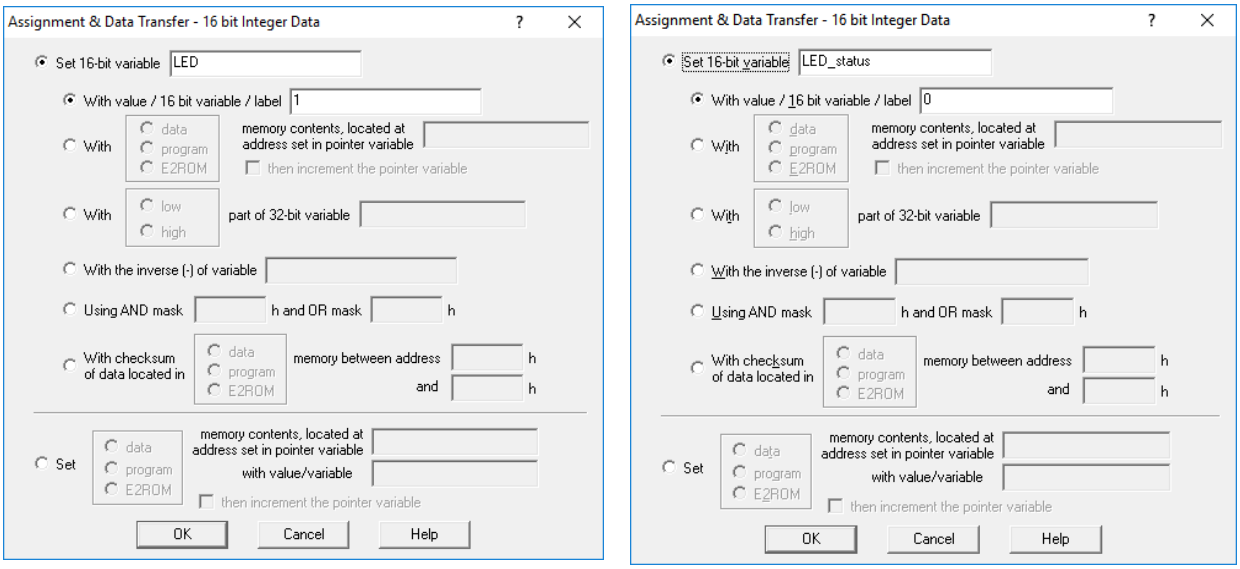

Figure 11. *Variable's value corresponding to the LED ON / OFF status*

• After the “OUT(0)” digital output state is changed and the “LED_status” variable is set accordingly, the program should return from the interrupt. This is done using the “RETI;” (return from interrupt) instruction. It can be inserted from the “Jumps and Function Calls” dialogue (1).

Remark: In in the second case (when the LED is switched off), the “RETI” instruction is not used anymore because the program returns naturally to the “Motion section” (the interrupt routine ends after “LED_status = 0” instruction).

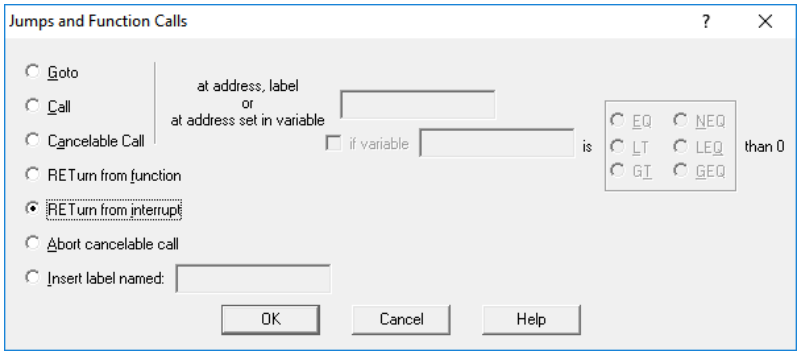

Figure 12. *How to insert a "RETI" TML instruction*

5. Application evaluation

This application requires to connect a LED to the "OUT(0)" digital output, according to the schematics in the drive user manual. If this is not possible, then the "2_Drive IO" control panel in EasyMotion Studio can be used.

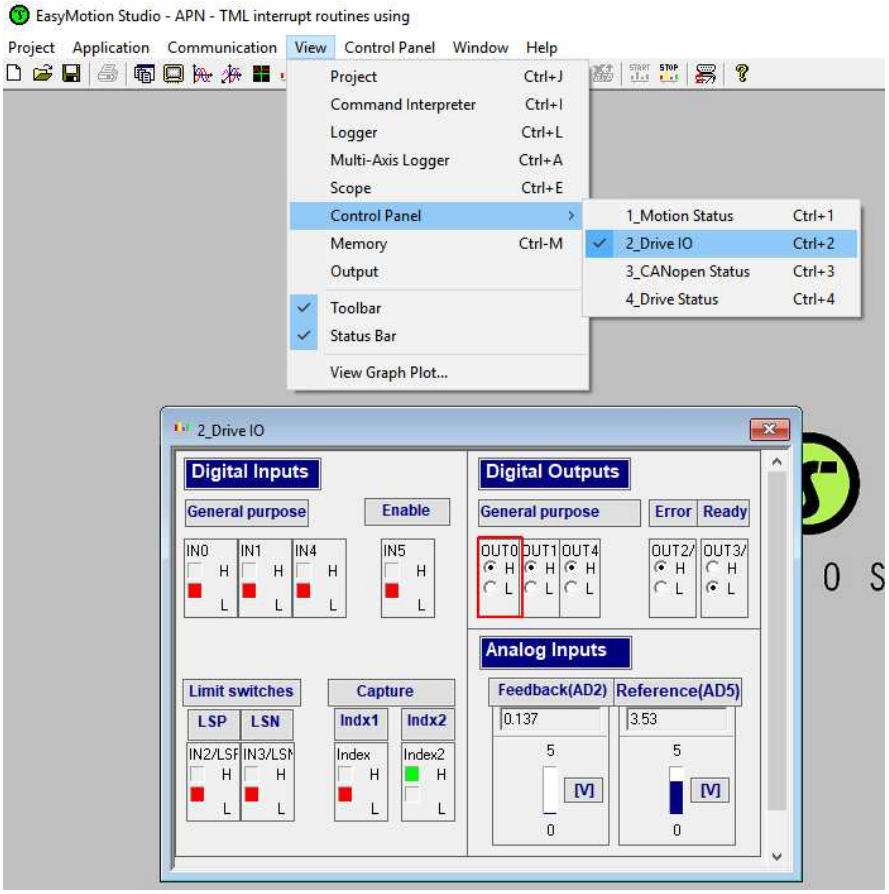

Figure 13. *How to insert a "RETI" TML instruction*

During the program execution the "OUT(0) digital output will switch High (H) and Low (L) each 0.5 respecting the algorithm presented in the previous chapter.