1. Application note description

The ENABLE input is continuous monitored by the drive firmware. When it switches to the active level, the drive PWM outputs are deactivated, the bit 15 in the MER error register (“Enable is inactive”) is set to 1 and the “int0 – Enable input has changed” interrupt routine is executed.

When the ENABLE input status changes to the inactive state, the bit 15 in the MER error register is reset but the PWM outputs remain deactivated. There is only one exception: if the drive was executing an electronic gearing profile, when the ENABLE input became active, then after the ENABLE input will switch back to the inactive state, the drive power stage and the electronic gearing mode will be enabled automatically.

Remark: The ENABLE input is available only on the drives without STO (safe torque off) functionality.

This application note describes how to program the drive, to reactivate the drive PWM outputs automatically, when the ENABLE input becomes inactive, no-matter what motion profile was in execution, when the ENABLE input changed to the active level.

This functionality was implemented using “int0 – Enable input has changed” interrupt routine. It was modified, to read the bit 15 (“Enable input is inactive”) in the MER error register. If it is 1, the axis is kept disabled. Otherwise, the drive PWM outputs are re-activated.

2. Application flow chart

![Diagram](image)

Figure 1. Enable TML interrupt structure & main motion structure
3. EasyMotion Studio implementation

Figure 2. Main application structure

Figure 3. Enable Interrupt routine structure
4. Detailed description

4.1 Motion sequences

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 (4), allows declaring the user variables and inserting different TML instructions. Here, it was used to declare the “MER_copy” user variable that will be used later, in the “int0 – Enable input has changed” interrupt service routine.

![Figure 4. How to declare a user variable](image)

• The “Interrupt Settings” dialog (5) allows to activate and/or deactivate the TML (Technosoft Motion Language) 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 chapter.

![Figure 5. How active the enable input](image)

• The “Jumps and Function Calls” dialogue (3) 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 a loop where the motor runs 5 rotations in the positive direction and then 5 rotations in the negative direction.

Figure 6. 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.

In this case it was used to move the motor for 5 rotations and then to return to 0.

Figure 7. How to configure and start motion using a trapezoidal position profile

• The “Events” dialogue (2) allows defining 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.

In this case, the “Events” dialog was used to insert a 0.3 s delay between the two movements. This reduces the shock when motion is reversed.

Figure 8. How to create an event
4.2 Enable input has changed 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.

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

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

- The “Assignment and Data Transfer – 16 bit Integer Data” dialogue (5) allows different operations with a 16 bit integer variable/parameter/register.

Here, the “Assignment and Data Transfer – 16 bit Integer Data” dialogue was used to set the “VAR_I1” internal variable with the value of the MER error register and then, to apply an “AND” and “OR” mask, that isolates the bit 15 in the MER error register (“Enable is inactive”).

![Figure 9. Operations using the “Assignment and Data Transfer – 16 bit Integer Data” dialog](image)

- The “Jumps and Function Calls” dialogue (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 10. Insert a “GOTO;” and “RETI;” TML instruction](image)
The “AXISENABLE” label was also created using the “Jumps and Function Calls” dialogue (4).

**Figure 11. How to create a label**

- The “Assignment and Data Transfer – 16 bit Integer Data” dialogue (5) was used again, to apply an “AND” and “OR” mask, that sets the bit 3 in the ASR register is set to 1. This will make the target speed 0, when the next position profile (with TUMO - 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 jumping to the last imposed position, when the drive power stage is reactivated.

**Figure 12. Set the bit 3 in the ASR register to 1**

- The “Motion – Trapezoidal Profiles” dialog (1) was used to execute a relative position profile, with “TUM0” and a position increment of 0 rot. This way the motor will hold the current position, after the error state disappears and the axis is re-enabled.

**Figure 13. How to configure and start motion using a position profile using TUM0**
• The “Motion - Motor Commands” dialogue (2) is used to reactivate the drive PWM outputs and allow the position profile above to start being executed.

![Motion - Motor Commands](image)

**Figure 14. How to set AXISON**

• 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.

![Events](image)

**Figure 15. Set an event on Motion Complete.**

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.