

Modeling and Automation of Industrial Processes

Modelação e Automação de Processos Industriais / MAPI

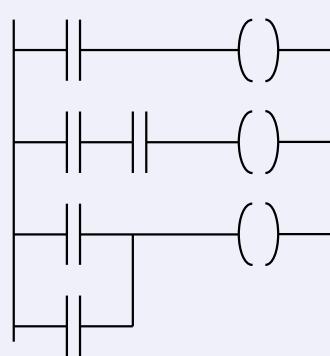
PLC Programming languages
Common Programming Errors

<http://www.isr.tecnico.ulisboa.pt/~jag/courses/mapi2223>

Prof. José Gaspar, rev. 2022/2023

PLC Programming Languages (IEC 61131-3)

Ladder Diagram



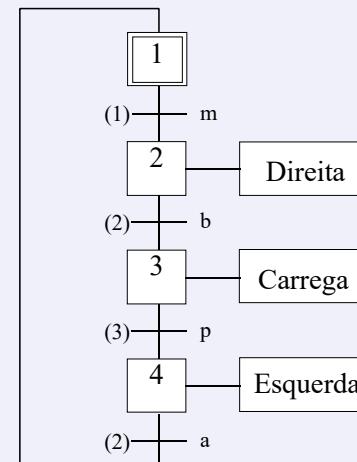
Structured Text

```
If %I1.0 THEN  
    %Q2.1 := TRUE  
ELSE  
    %Q2.2 := FALSE  
END_IF
```

Instruction List

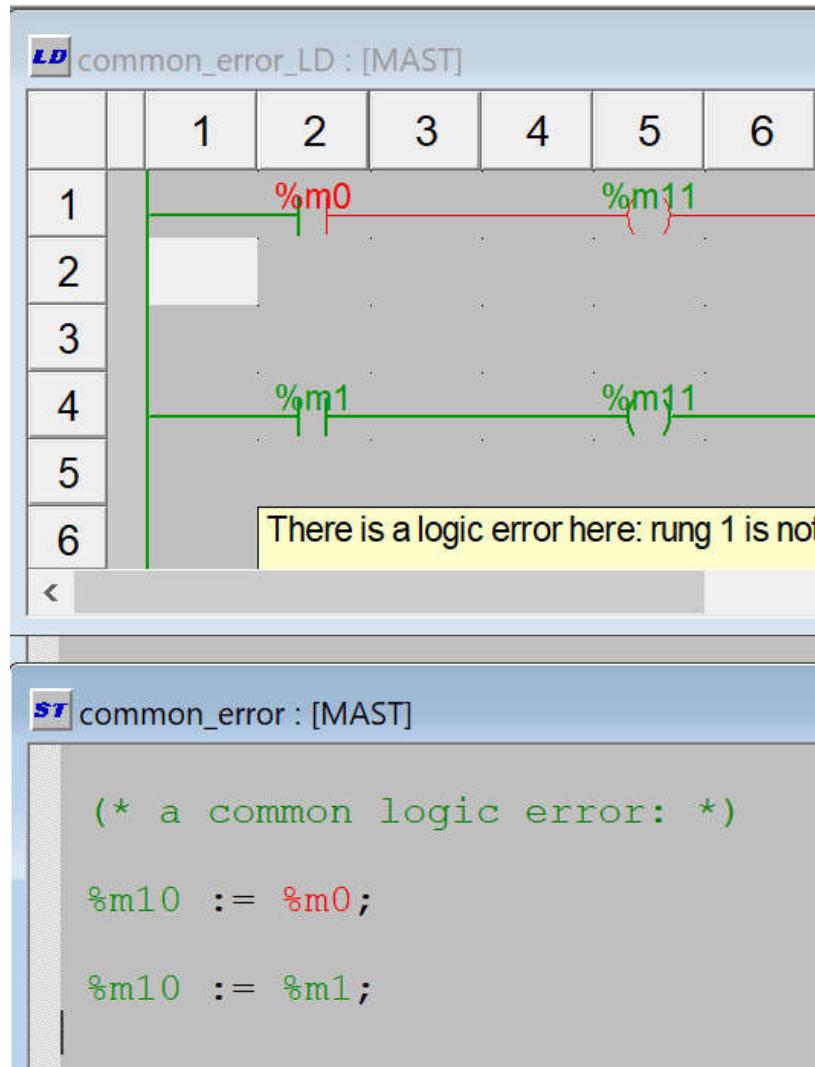
LD	%M12
AND	%I1.0
ANDN	%I1.1
OR	%M10
ST	%Q2.0

Sequential Function Chart (GRAFCET)

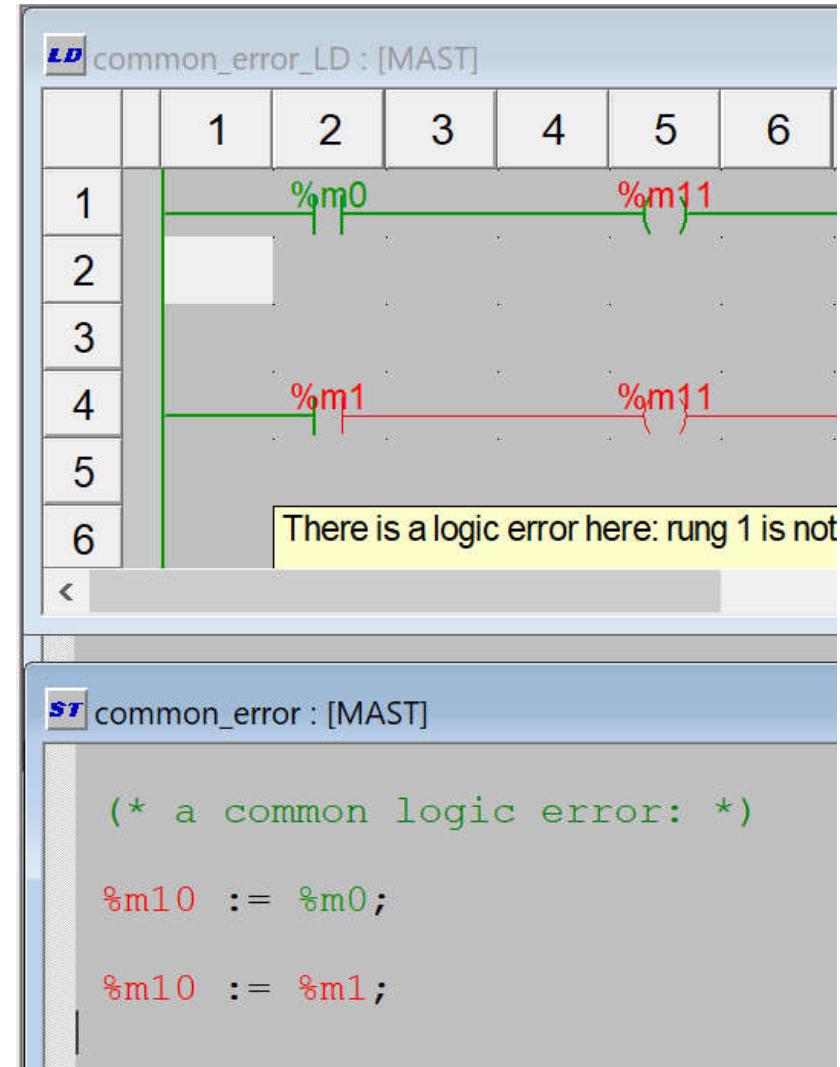


*1. Multiple writes to one output in
the same scan cycle*

A very common programming error:



Noting `%m0` is FALSE
why do we have `%m11` and `%m10 = TRUE`?



Noting `%m0` is TRUE
why do we have `%m10` and `%m11 = FALSE`?

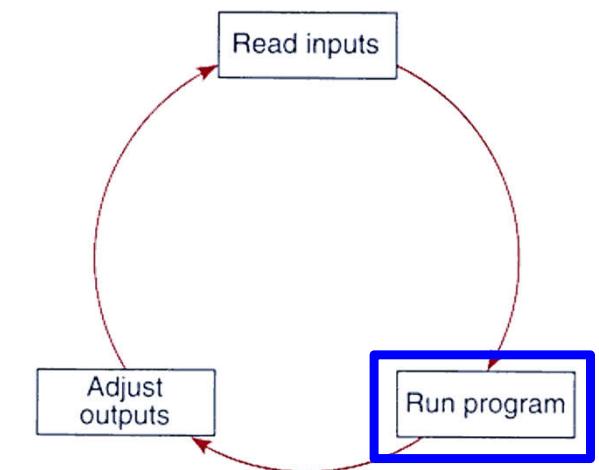
A very common programming error:

The screenshot shows a Project Browser on the left and two code editors on the right.

- Project Browser:**
 - Structural view:** Shows Configuration, Derived Data Types, Derived FB Types, Variables & FB instances, Motion, Communication, and Program.
 - Program section:** Under Tasks > MAST > Sections, three sections are listed: set_kb_cols, common_error, and common_error LD. The common_error LD section is highlighted with a blue box.
- Logic Editor (LD):** The common_error_LD section contains a ladder logic diagram with six rungs. Rung 1 has contacts %m0 and %m1 in series, leading to coil %m11. Rung 4 has contact %m1 in series, leading to coil %m11. A yellow message box states: "There is a logic error here: rung 1 is no".
- Text Editor (ST):** The common_error section contains the following code:

```
(* a common logic error: *)
%m10 := %m0;
%m10 := %m1;
```

All this code is executed in 1 single scan cycle



Real outputs are what the hardware connected to the PLC see and what **you see on screen**.

Detail: The first assignment
 $\%m10 := \%m0;$
is overwritten by the second
 $\%m10 := \%m1;$

*2. Timers in subroutines not running
are not reset*

The screenshot shows a software interface for PLC programming. On the left is the 'Project Browser' window with a 'Structural view' tree containing various project components like Configuration, Derived Data Types, Derived FB Types, Variables & FB instance, Motion, Communication, Program (Tasks, MAST, Sections, SR Sections, Events), Animation Tables, Operator Screens, and Documentation.

The main area contains four code editors:

- s1 : [MAST]**: Shows a TON timer definition and a call to sub_1. The TON definition is highlighted with a blue box around the output coil Q => %m10.
- sub_1 <SR> : [MAST]**: Shows a TON timer definition for %m11. The output coil Q => %m11 is also highlighted with a blue box.
- sub_2 <SR> : [MAST]**: Shows a TON timer definition for %m12. The output coil Q => %m12 is highlighted with a blue box.

```

(* timeout 1sec after RE(%m0) : *)
TON_0 (IN := %m0 (*BOOL*),
       PT := t#1s (*TIME*),
       Q => %m10 (*BOOL*) );
(* a timeout makes %m10 True
   False %m0 makes %m10 False *)

sub_1 (); (* CALL sub_1() *)

if %m0 then
  sub_2 (); (* CALL sub_2() *)
end_if;

TON_1 (IN := %m0 (*BOOL*),
       PT := t#1s (*TIME*),
       Q => %m11 (*BOOL*) );
(* a timeout makes %m11 True
   False %m0 makes %m11 False *)

TON_2 (IN := %m0 (*BOOL*),
       PT := t#1s (*TIME*),
       Q => %m12 (*BOOL*) );
(* after timeout, why False %m0
   does NOT make %m12 False ? *)

```

False %m0 is making false %m11, as expected.

However, we also see

False %m0 and True %m12. Why is %m12 True?

One timer can be called multiple times

```
(* After declaring a timer in FB instances, PT needs to be set.  
Setup timer to start on %M0 and timeout on %M10.  
Note: no need to include arg ET of type TIME.  
*)  
TON_0 (IN := %m0 (*BOOL*),  
       PT := t#3s (*TIME*),  
       Q   => %m10 (*BOOL*));  
  
(* Use timer to timeout also on %m11 *)  
TON_0 (Q => %m11 (*BOOL*));  
  
(* Use timer name "as a structure" *)  
%m12 := TON_0.Q;  
  
(* Use a separate call to reset timer *)  
if %m1 then  
    TON_0( IN := False );  
end_if;  
  
(* Auto reset if %m3 is True. Use it to toggle %M13. *)  
if %m3 AND %m10 then  
    TON_0( IN := False );  
    %m13 := NOT(%m13);  
end_if;  
  
(* Use a separate call to redefine PT *)  
if %m2 then  
    TON_0( PT := t#1s );  
end_if;
```