

Modeling and Automation of Industrial Processes

Modelação e Automação de Processos Industriais / MAPI

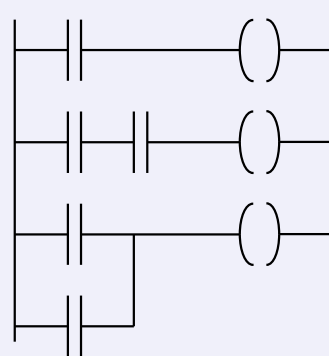
PLC Programming languages ***Common Programming Errors***

<http://www.isr.tecnico.ulisboa.pt/~jag/courses/mapi2122>

Prof. Paulo Jorge Oliveira, original slides
Prof. José Gaspar, rev. 2021/2022

PLC Programming Languages (IEC 61131-3)

Ladder Diagram



Structured Text

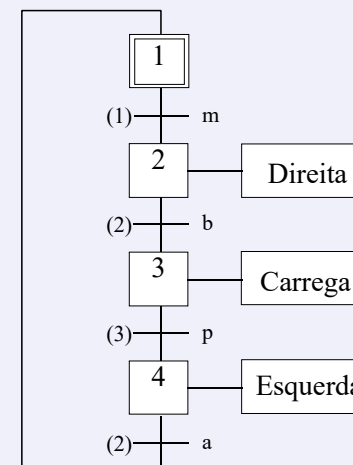
```

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

Instruction List

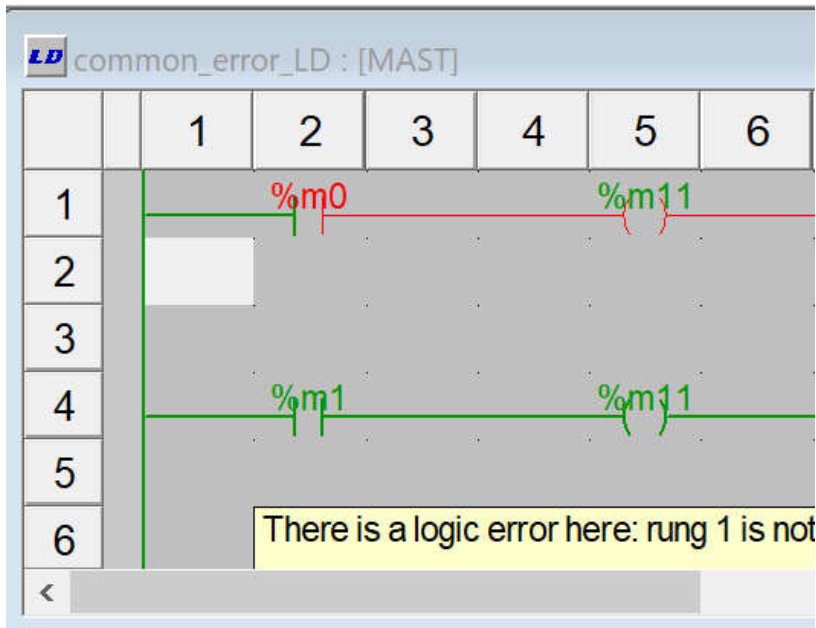
LD	%M12
AND	%I1.0
ANDN	%I1.1
OR	%M10
ST	%Q2.0

Sequential Function Chart (GRAFCET)



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

A very common programming error:

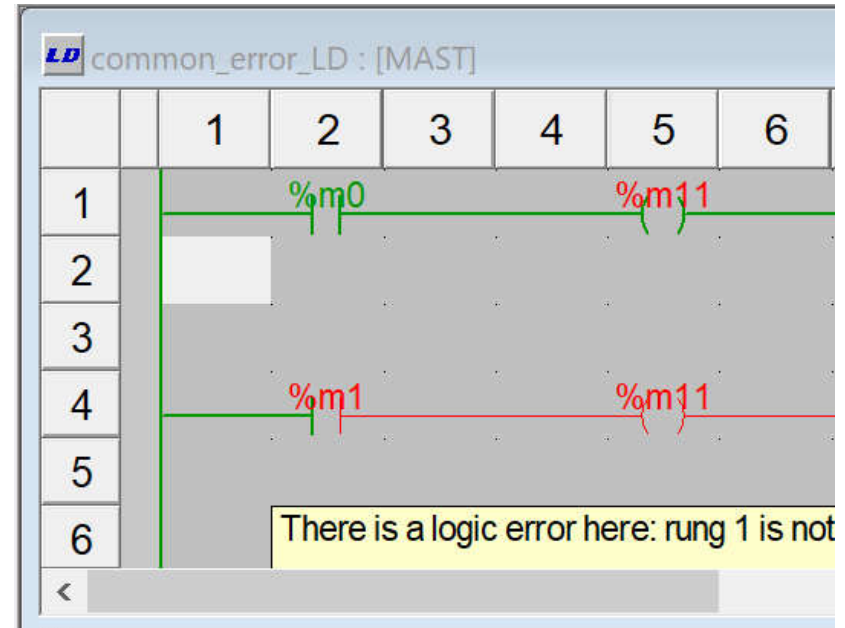


```

LD common_error_LD : [MAST]

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

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



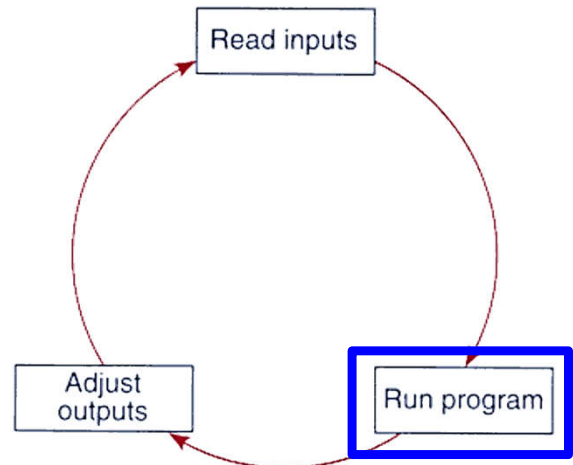
```

LD common_error_LD : [MAST]

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

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

A very common programming error:



LD common_error_LD : [MAST]

	1	2	3	4	5	6
1	%m0		%m11			
2						
3						
4	%m1		%m11			
5						
6						

There is a logic error here: rung 1 is no

ST common_error : [MAST]

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

Adjusted outputs is what the hardware connected to the PLC see as IO and also what you see on screen.

*Note: The first assignment
%m10 := %m0;
is overwritten by the second
%m10 := %m1;*

Code to run in 1 single scan cycle

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

Project Browser

- Structural view
- Project
 - Configuration
 - Derived Data Types
 - Derived FB Types
 - Variables & FB instance
 - Motion
 - Communication
 - Program
 - Tasks
 - MAST
 - Sections
 - s1
 - SR Sections
 - sub_1
 - sub_2
 - Events
 - Animation Tables
 - Operator Screens
 - Documentation

```
ST s1 : [MAST]
(* 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;
```

```
ST sub_1 <SR> : [MAST]
TON_1 (IN := %m0 (*BOOL*),
      PT := t#1s (*TIME*),
      Q => %m11 (*BOOL*) );

(* a timeout makes %m11 True
   False %m0 makes %m11 False *)
```

```
ST sub_2 <SR> : [MAST]
TON_2 (IN := %m0 (*BOOL*),
      PT := t#1s (*TIME*),
      Q => %m12 (*BOOL*) );

(* after timeout, why False %m0
   does NOT make %m12 False ? *)
```

How is it possible %m12 is 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;
```