


Sequential Function Chart

CoDeSys

Smart Software Solutions

We software Automation.




Sequential Function Chart

is a graphically oriented language which allows to describe the chronological order of particular actions within a program

Smart Software Solutions

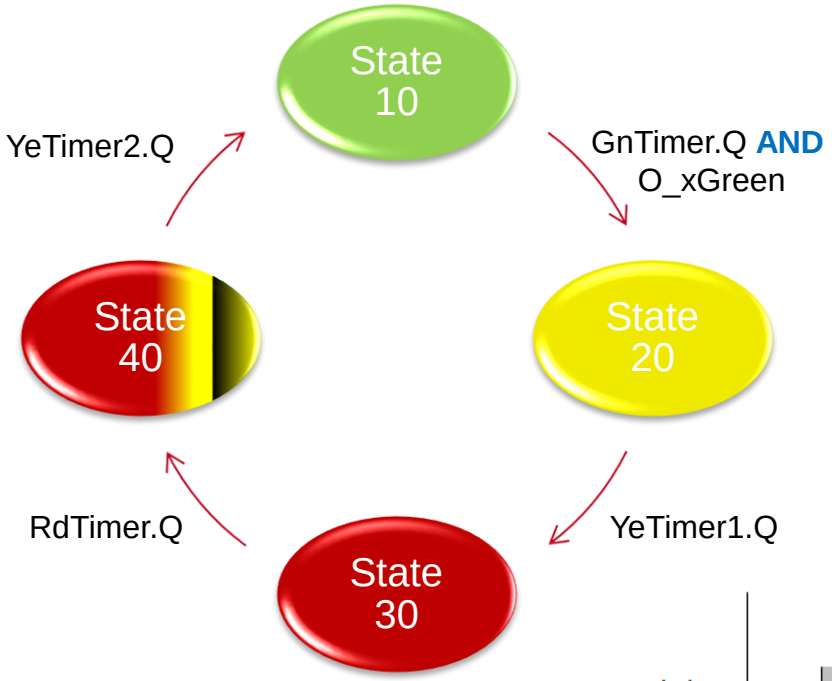
We software Automation.

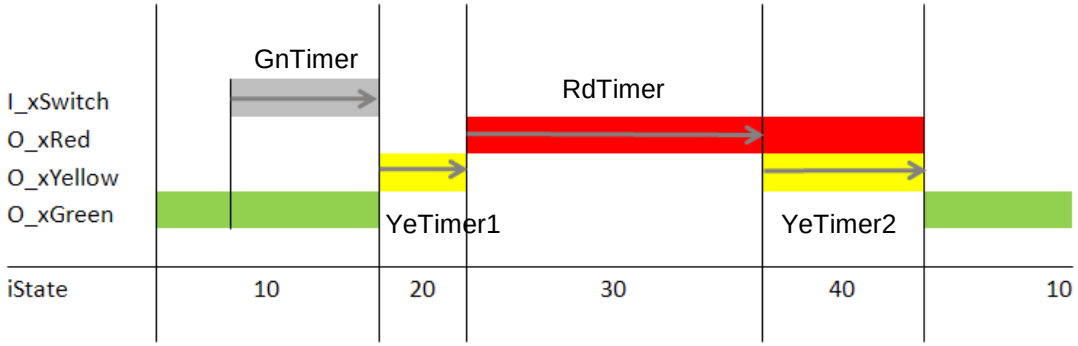
Introduction

- After this module..
  - you will be familiar with the new graphical language and
  - you will be able to read and write POUs in SFC.

Review Project “TFL in ST”

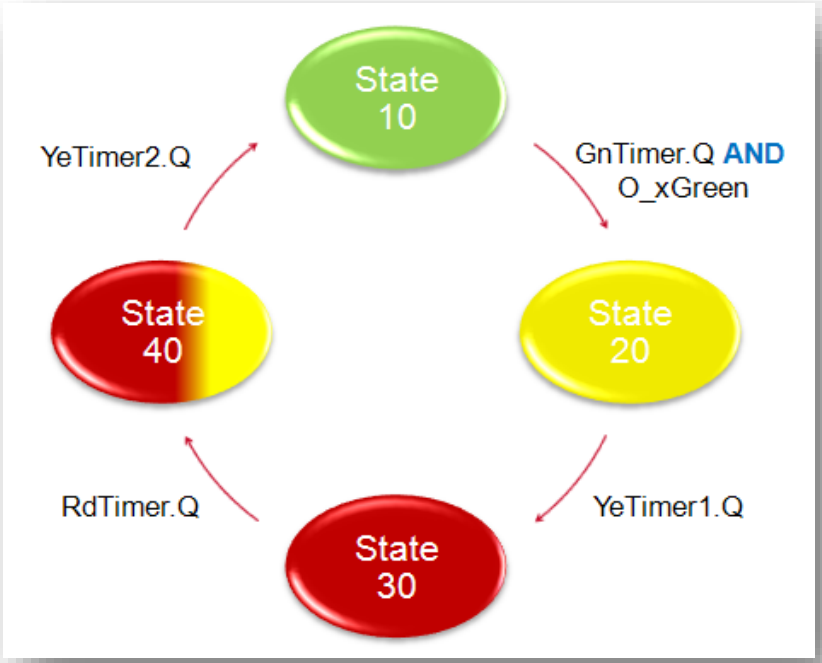
Project was realized with a State Machine in ST





Review Project “TFL in ST”

State Machine

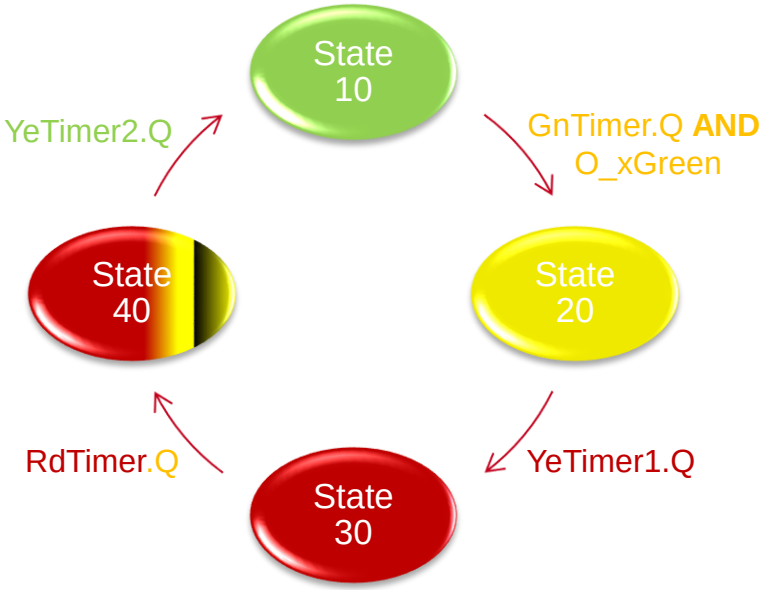


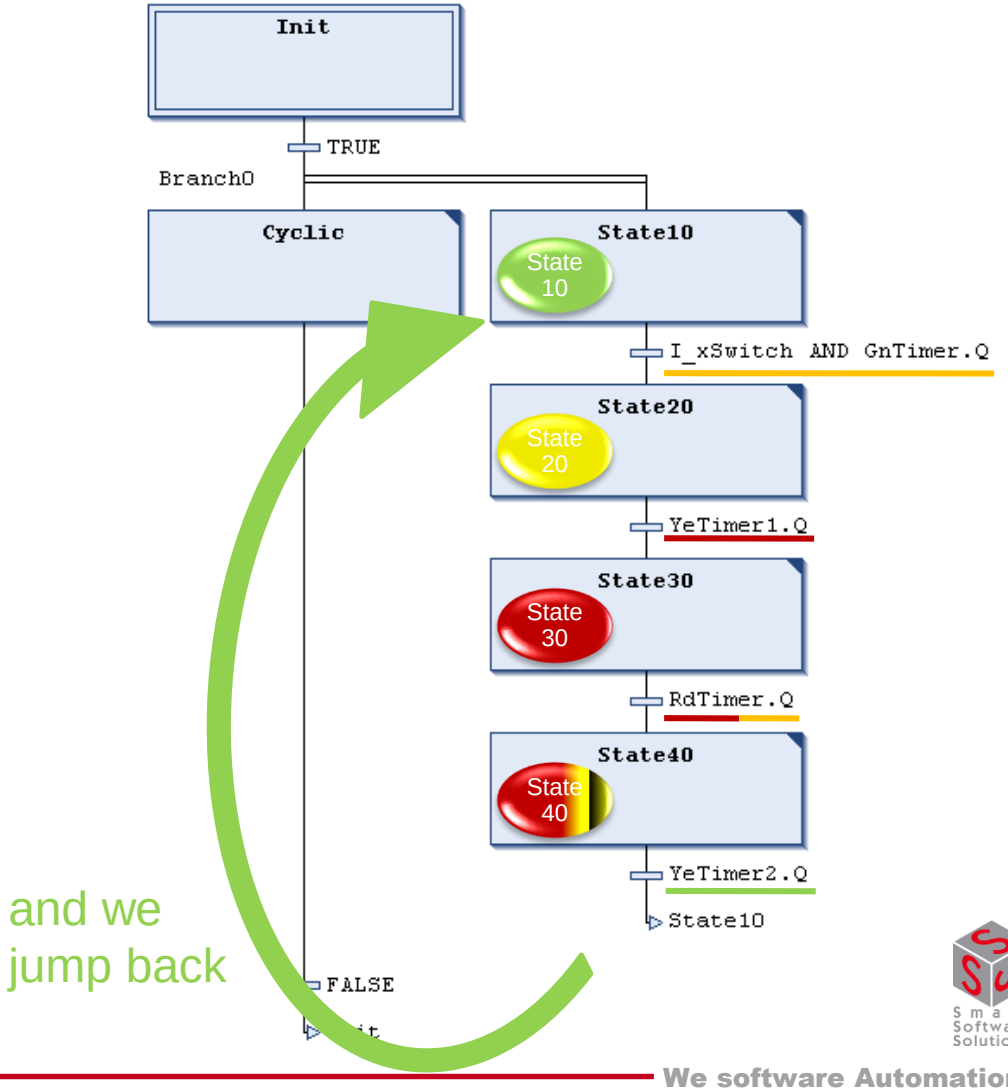

```
CASE iState OF
    10:
        O_xRed := FALSE;
        O_xYellow := FALSE;
        O_xGreen := TRUE;

        IF I_xSwitch AND GnTimer.Q THEN
            iState := 20;
        END_IF
    20:
        O_xRed := FALSE;
        O_xYellow := TRUE;
        O_xGreen := FALSE;

        IF YeTimer1.Q THEN
            iState := 30;
        END_IF
    30:
        O_xRed := TRUE;
        O_xYellow := FALSE;
        O_xGreen := FALSE;

        IF RdTimer.Q THEN
            iState := 40;
        END_IF
    40:
        O_xRed := TRUE;
        O_xYellow := TRUE;
        O_xGreen := FALSE;

        IF YeTimer2.Q THEN
            iState := 10;
        END_IF
END_CASE
```

Project Conversion

Let‘s program the traffic light in SFC

- State Machine



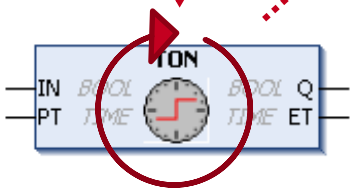


Project Conversion

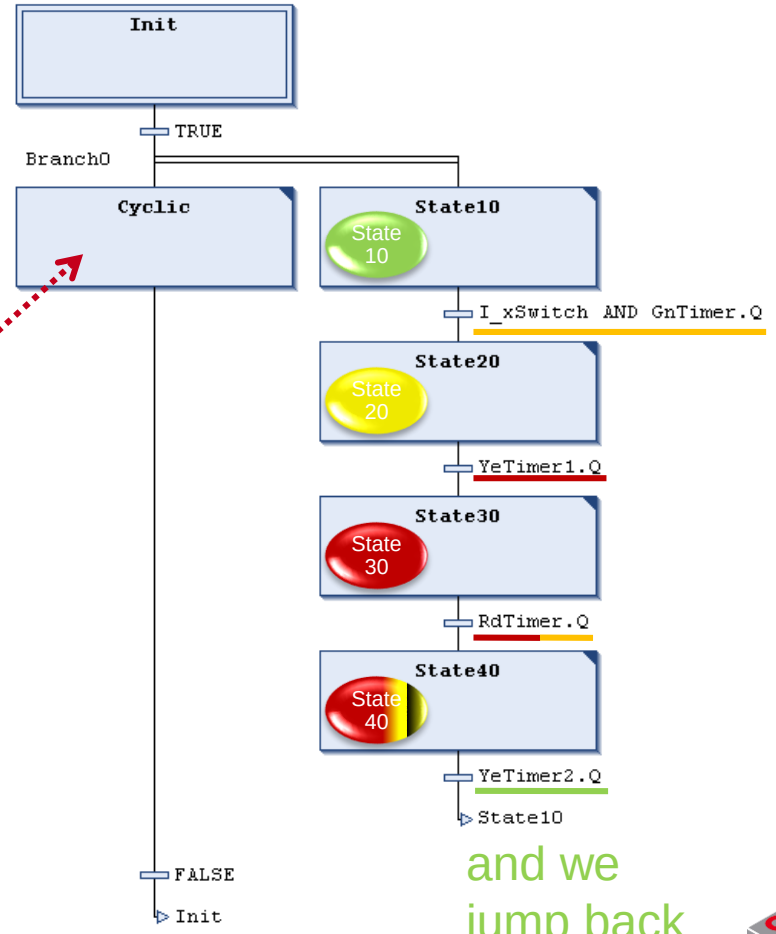
Let‘s program the traffic light in SFC

- cyclic actions in parallel branch

```
PROGRAM MainPLC
VAR
    iState      : INT := 10;

    GnTimer     : TON;
    YeTimer1    : TON;
    YeTimer2    : TON;
    RdTimer     : TON;
END_VAR
```



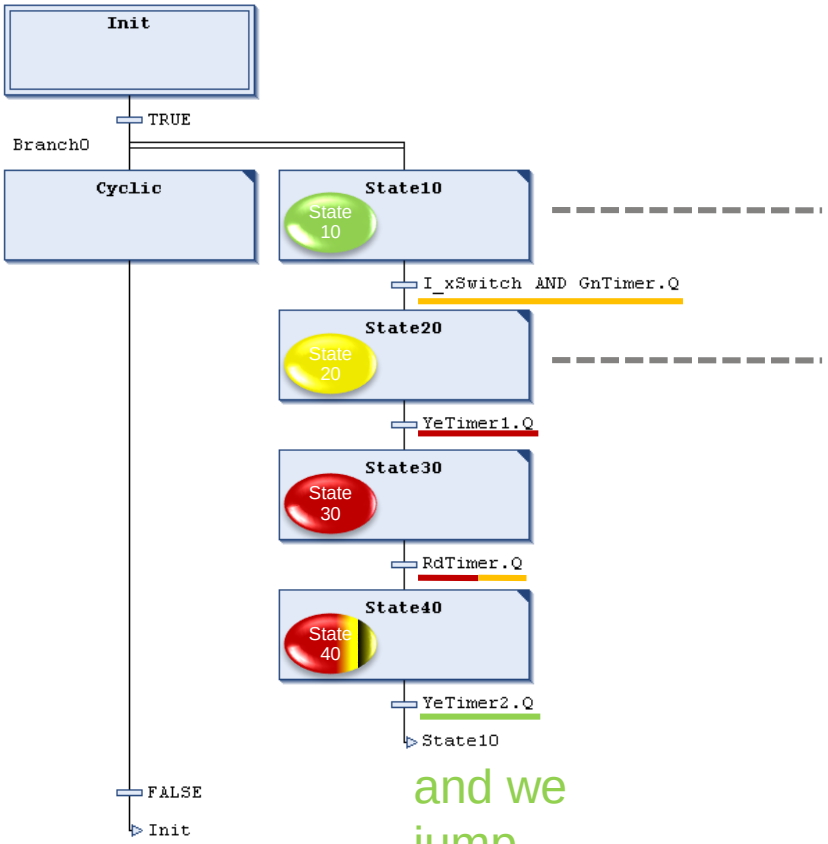
don‘t forget to call your instance in every cycle



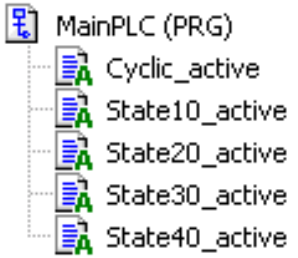
and we jump back

Let‘s program the traffic light in SFC

- A step can have actions



and we jump back



State10_active [PLC1: PLC Logic: Application: MainPLC]

```
1  O_xRed := FALSE;
2  O_xYellow := FALSE;
3  O_xGreen := TRUE;
```

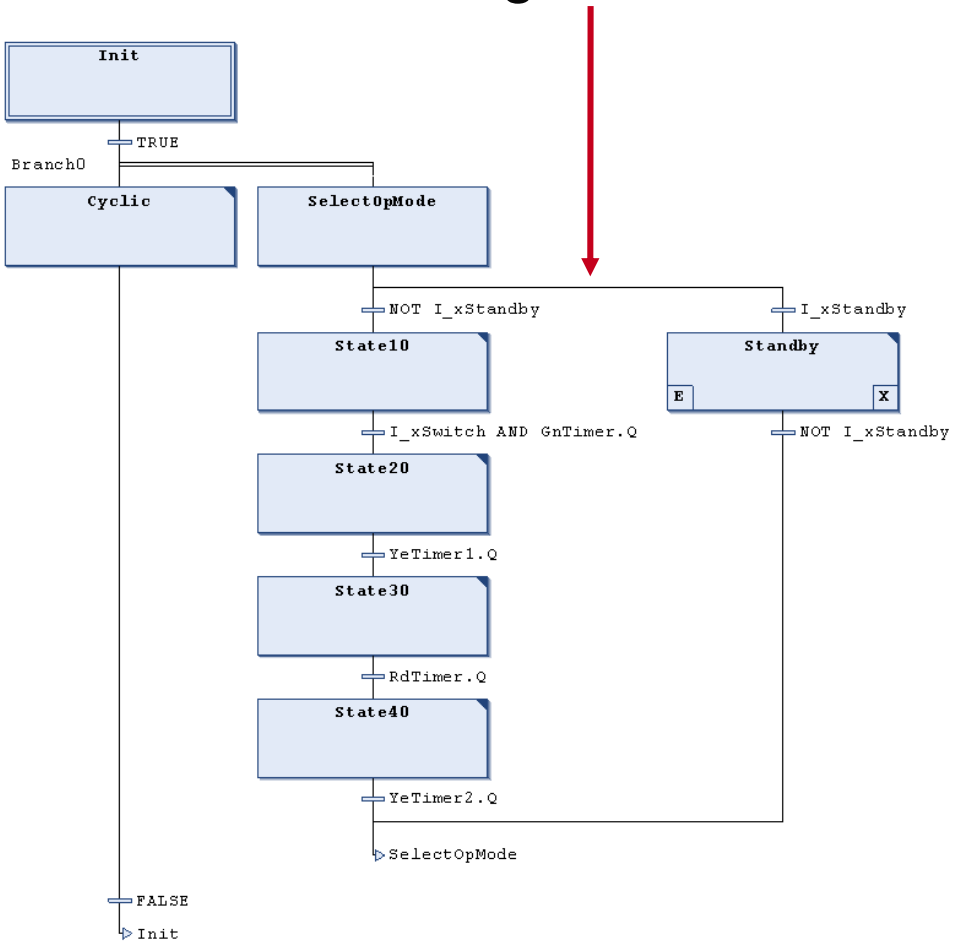
State20_active [PLC1: PLC Logic: Application: MainPLC]

```
1  O_xRed := FALSE;
2  O_xYellow := TRUE;
3  O_xGreen := FALSE;
```

Branching

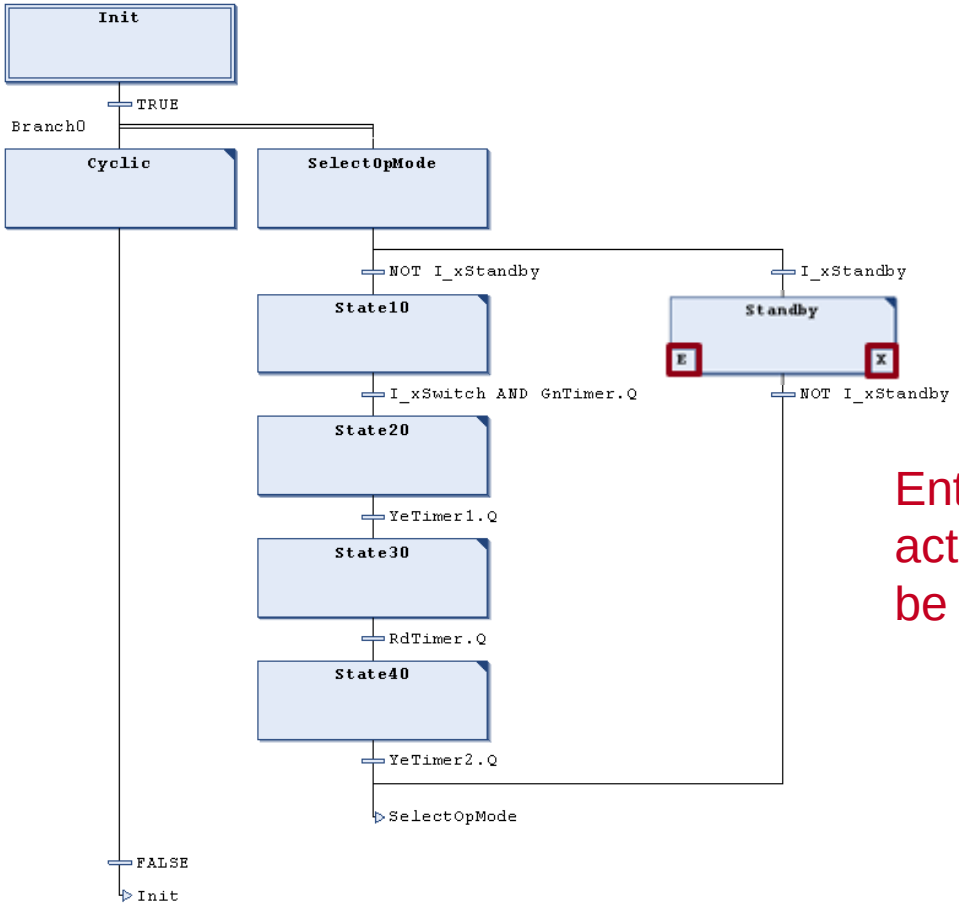
Parallel and alternative branching is possible

- Alternative branches have a single line



One step can have several actions

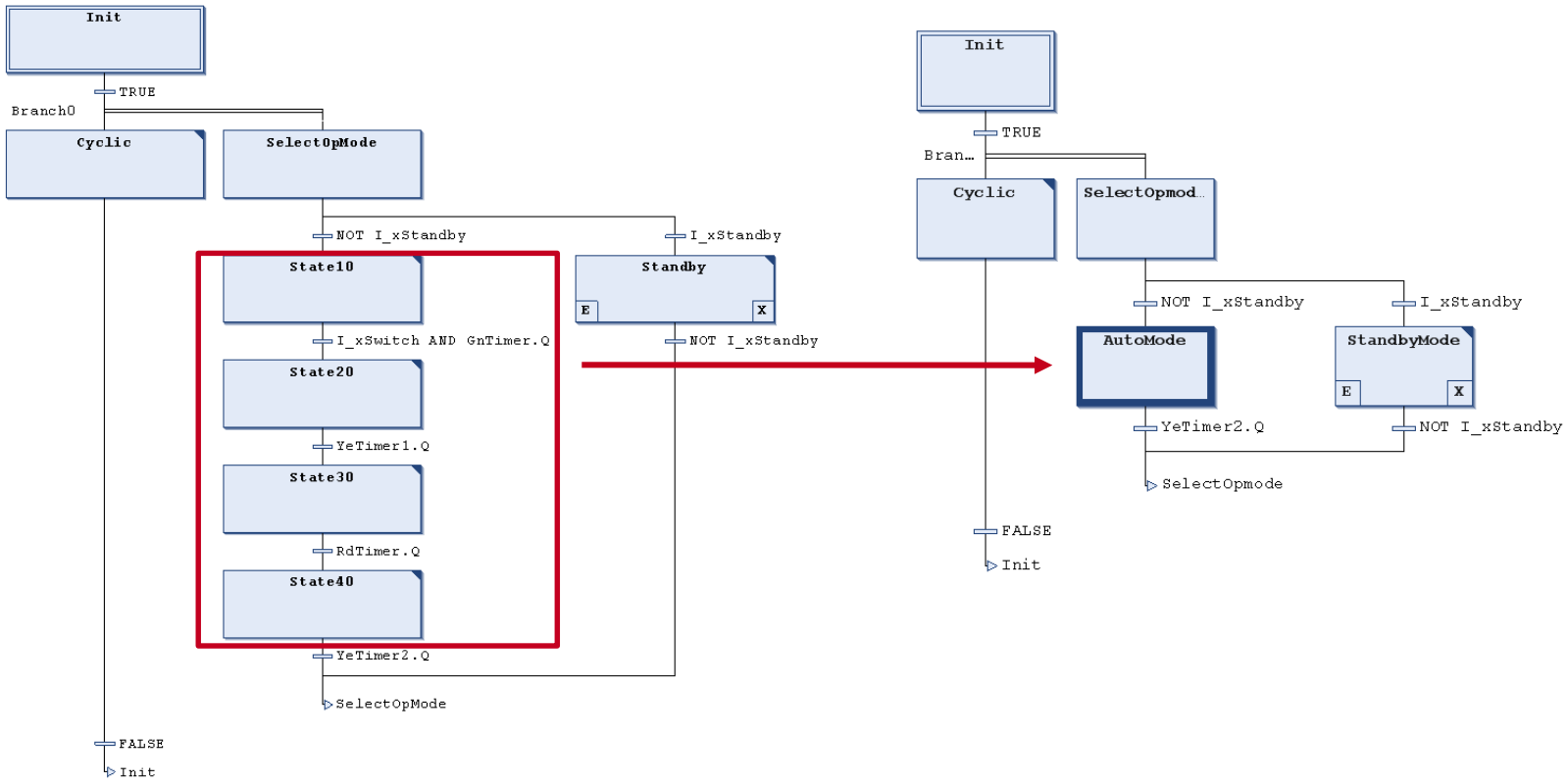
- Main, entry and exit actions are possible



Entry and Exit actions will only be executed once.

You can put several elements into a macro

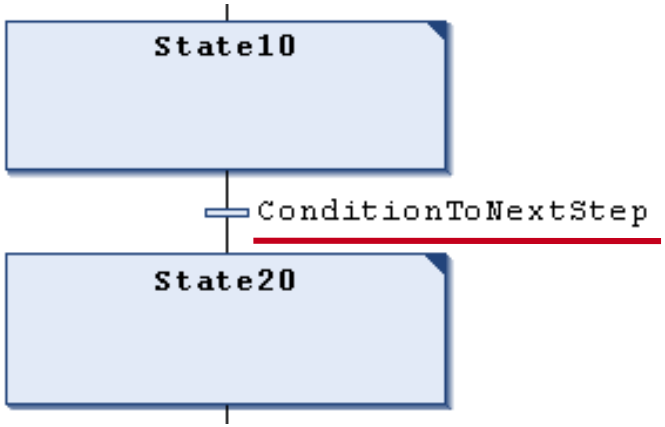
- For example the steps of the normal operation

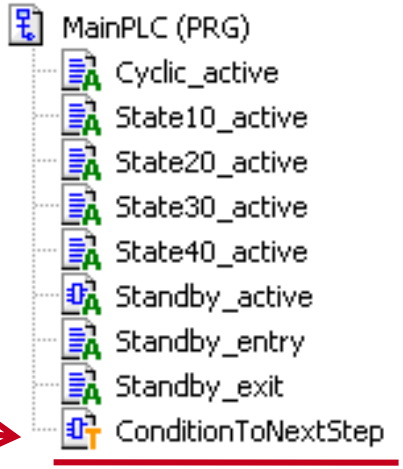


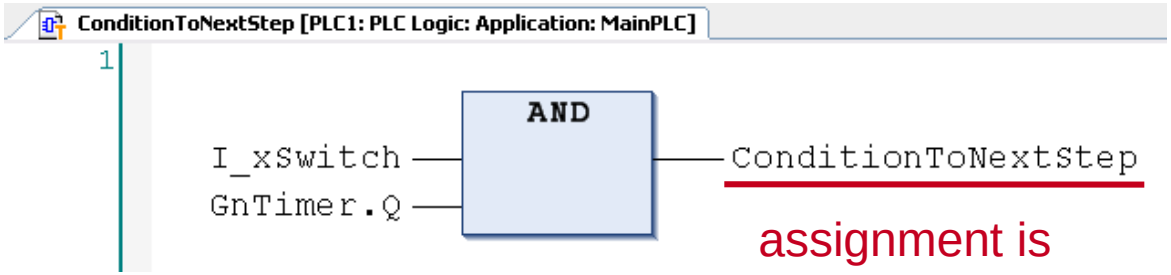
Project Conversion

A transition can be written in another language

- add a transition in FBD





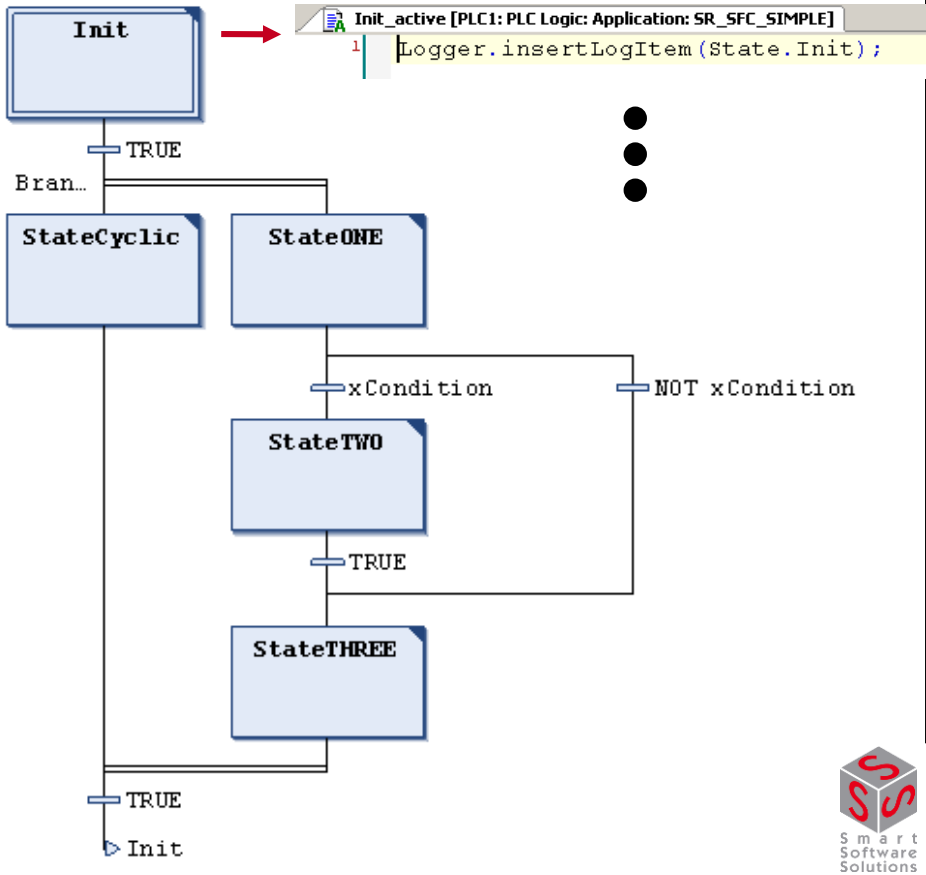


assignment is optional

How the SFC POU is executed

- in comparison to ST

```
1  Logger.insertLogItem(State.Init);
2
3  Logger.insertLogItem(State.Cyclic);
4  Logger.insertLogItem(State.ONE);
5
6  IF xCondition THEN
7      Logger.insertLogItem(State.TWO);
8  END_IF
9
10 Logger.insertLogItem(State.THREE);
11
```



How the SFC POU is executed

- execution in ST

```
1  Logger.insertLogItem(State.Init);
2
3  Logger.insertLogItem(State.Cyclic);
4  Logger.insertLogItem(State.ONE);
5
6  IF xCondition THEN
7      Logger.insertLogItem(State.TWO);
8  END_IF
9
10 Logger.insertLogItem(State.THREE);
```

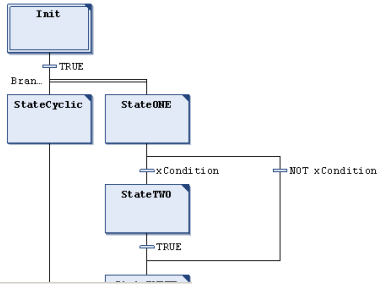
Vis_ST

State
NewPlcCycle
Init
Cyclic
ONE
THREE
NewPlcCycle
Init
Cyclic
ONE
THREE
NewPlcCycle
Init
Cyclic
ONE
THREE
NewPlcCycle

PLCCycle / Counter State Executed

	NewPLCCycle	Cyclic	Init	Zero	One	Two	Three
0	0	0	0	0	0	0	0
1	1	1	1	0	1	0	1
2	2	2	2	0	2	0	2
3	3	3	3	0	3	0	3
4	4	4	4	0	4	0	4
5	5	5	5	0	5	0	5
6	6	6	6	0	6	0	6
7	7	7	7	0	7	0	7
8	8	8	8	0	8	0	8
9	9	9	9	0	9	0	9
10	10	10	10	0	10	0	10
11	11	11	11	0	11	0	11
12	12	12	12	0	12	0	12
13	13	13	13	0	13	0	13
14	14	14	14	0	14	0	14
15	15	15	15	0	15	0	15

How the SFC POU is executed



- execution in SFC (3S version)

We need 3 plc cycles to do the same.

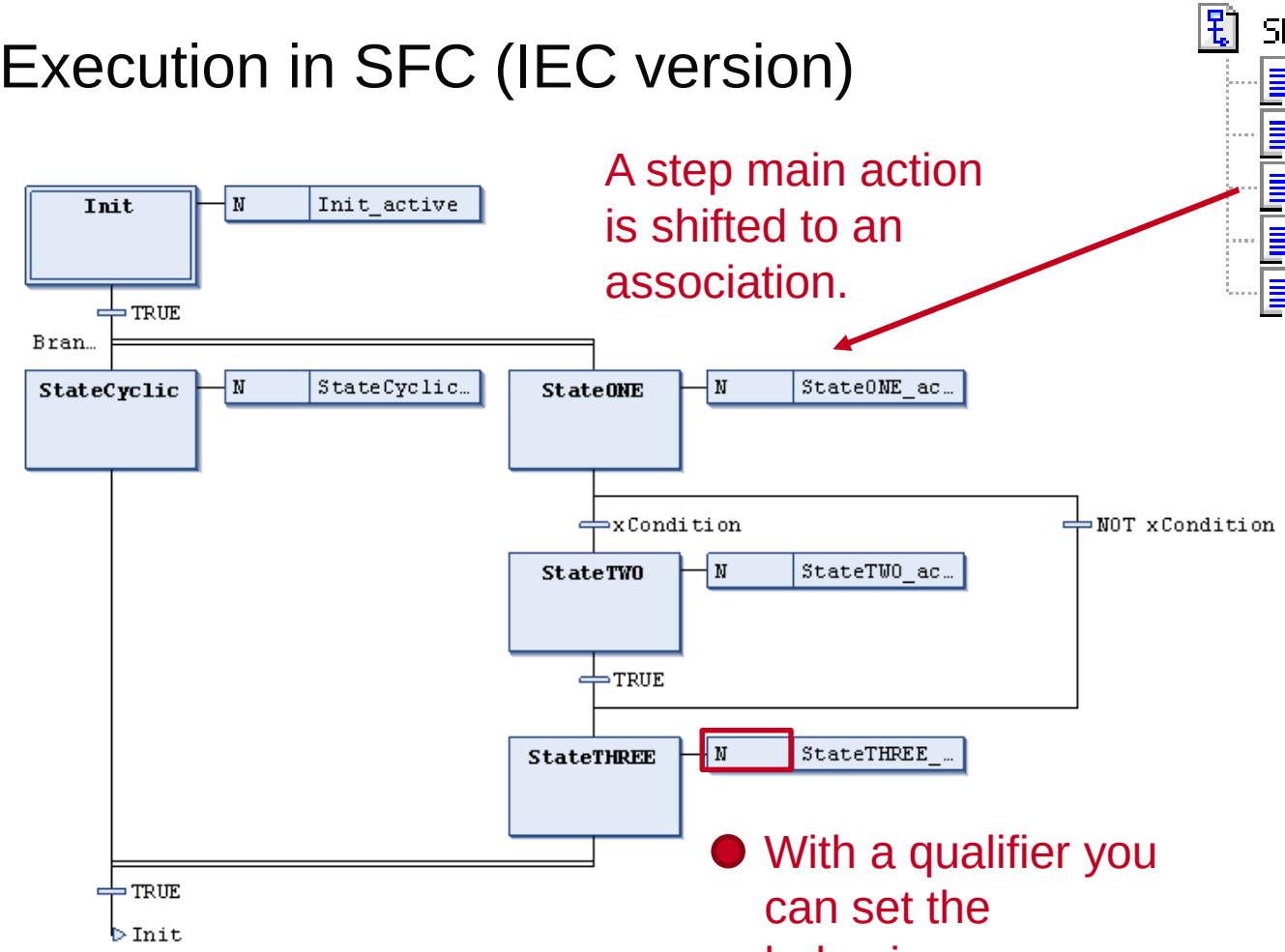
State
NewPlcCycle
Init
NewPlcCycle
Cyclic
ONE
NewPlcCycle
Cyclic
THREE
NewPlcCycle
Init
NewPlcCycle
Cyclic
ONE
NewPlcCycle
Cyclic
THREE
NewPlcCycle
Init
NewPlcCycle

PLCCycle / Counter State Executed

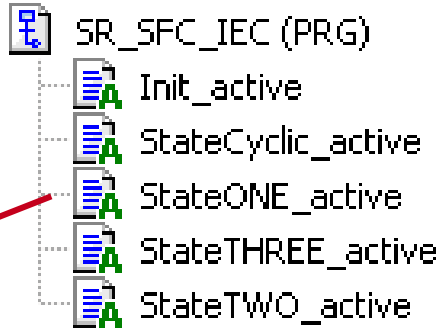
	NewPLCCycle	Cyclic	Init	Zero	One	Two	Three
0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0
2	2	1	1	0	1	0	0
3	3	2	1	0	1	0	1
4	4	2	2	0	1	0	1
5	5	3	2	0	2	0	1
6	6	4	2	0	2	0	2
7	7	4	3	0	2	0	2
8	8	5	3	0	3	0	2
9	9	6	3	0	3	0	3
10	10	6	4	0	3	0	3
11	11	7	4	0	4	0	3
12	12	8	4	0	4	0	4
13	13	8	5	0	4	0	4
14	14	9	5	0	5	0	4
15	15	10	5	0	5	0	5
16	16	10	6	0	5	0	5
17	17	11	6	0	6	0	5
18	18	12	6	0	6	0	6

How the SFC (IEC) POU is executed

- Execution in SFC (IEC version)



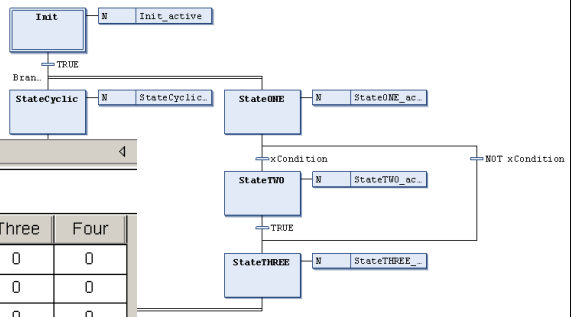
A step main action is shifted to an association.



- With a qualifier you can set the behaviour.

How the SFC (IEC) POU is executed

- Execution in SFC (IEC version)



We need 7 plc cycles to do the same.

Vis_SFC_IEC

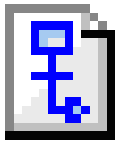
State
NewPlcCycle
Init
NewPlcCycle
Init
Cyclic
ONE
NewPlcCycle
ONE
Cyclic
THREE
NewPlcCycle
Cyclic
THREE
Init
NewPlcCycle
Init
Cyclic
ONE
NewPlcCycle
ONE
Cyclic
THREE
NewPlcCycle
Cyclic
THREE
Init
NewPlcCycle

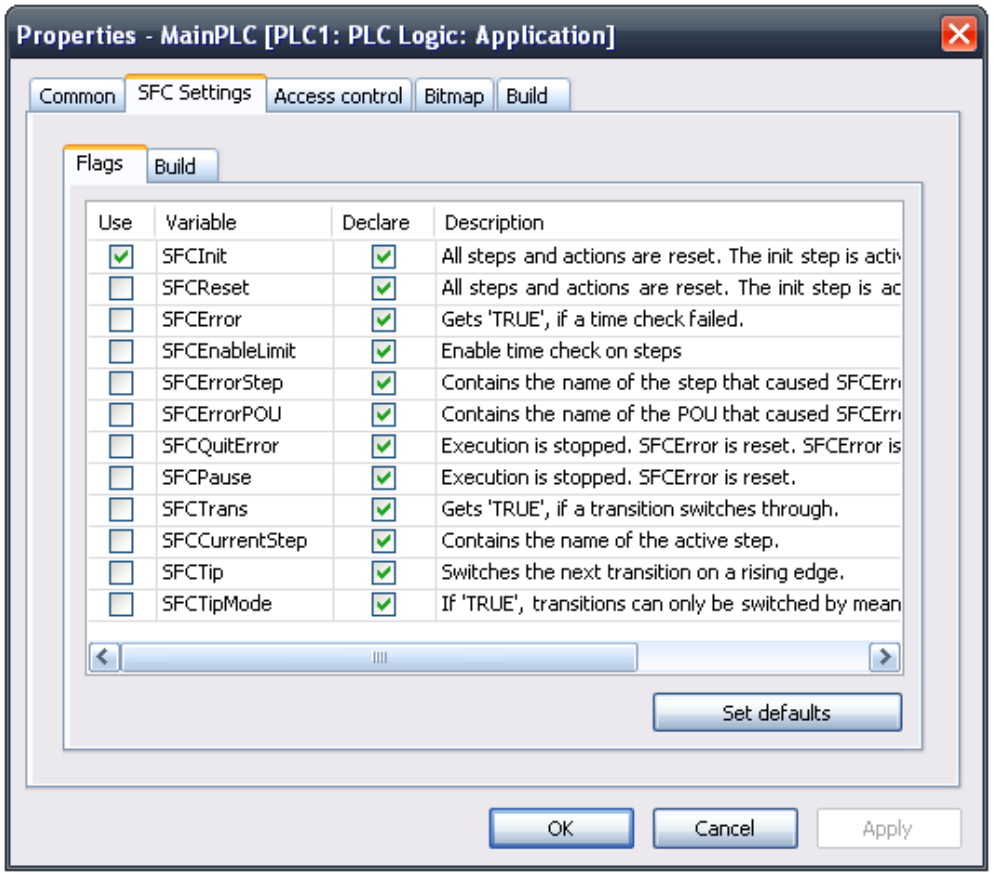
PLCCycle / Counter State Executed

	NewPLCCycle	Cyclic	Init	Zero	One	Two	Three	Four
0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0
2	2	1	2	0	1	0	0	0
3	3	2	2	0	2	0	1	0
4	4	3	3	0	2	0	2	0
5	5	4	4	0	3	0	2	0
6	6	5	4	0	4	0	3	0
7	7	6	5	0	4	0	4	0
8	8	7	6	0	5	0	4	0
9	9	8	6	0	6	0	5	0
10	10	9	7	0	6	0	6	0
11	11	10	8	0	7	0	6	0
12	12	11	8	0	8	0	7	0
13	13	12	9	0	8	0	8	0
14	14	13	10	0	9	0	8	0
15	15	14	10	0	10	0	9	0
16	16	15	11	0	10	0	10	0
17	17	16	12	0	11	0	10	0
18	18	17	12	0	12	0	11	0
19	19	18	13	0	12	0	12	0
20	20	19	14	0	13	0	12	0
21	21	20	14	0	14	0	13	0
22	22	21	15	0	14	0	14	0
23	23	22	16	0	15	0	14	0
24	24	23	16	0	16	0	15	0
25	25	23	16	0	16	0	15	0

Implicit Variables

- In the property dialog you will find all implicit variables.





Summary

- What is the typical structure of an SFC POU or action?
- What kinds of actions can you add to a step?
- How can you use implicit variables?