

CPDev – SFC programming

CONTENTS

• Introduction	1
• Greenhouse	1
Control problem	1
State diagram	1
SFC diagram prototype	2
• New program	2
Project.....	2
Program name and language.....	3
• SFC graphic editor	3
Main window	3
Editor options.....	5
SFC library.....	6
Step monitoring.....	6
• Development steps	7
Global variables.....	7
Initial SFC diagram	7
Transitions and actions.....	7
Task, Compilation, Simulation	7
• Initial SFC diagram	8
Main branch.....	8
Sequence selection	8
Jumps	9
Verification.....	10
Renumbering	10
• Conditions and actions	12
Transition conditions.....	12
Actions	13
Binding actions to steps.....	14
Verification and ST code	15
• Task and compilation	16
Task	16
Compilation – Build.....	17
• Online mode	13
Go online	13
SFC diagram online	13
Value list	15
Data sources	15
• CPSim simulator	19
Global variables.....	19
Control panel	20
• Diagram corrections	21
Selecting elements	21
Deleting single elements	21
Selecting pairs	22
Deleting pairs.....	23

• Reverse switching	24
Control problem	24
Preliminary diagrams	24
Initial SFC diagram	24
Conditions and actions	26
Actions	27
• SFC diagram	28
Simulation	28
• Tanks	29
Control problem	29
Tanks_SFC project	30
• Initial diagram	31
Development steps	31
First section	31
Simultaneous sequence	31
Sequence selection branches	32
Third section	34
Renumbering	34
• Completing the project	35
Conditions and actions	35
Volumes in tanks	37
Simulation	38
• SFC editor summary	40
Elements and evolution rules	40
SFC_basics	40
Local variables	41
Actions	42
Action blocks	42
Editing SFC diagram	42
Exchange and move	43
Unreachable and unsafe SFC	45
• Properties windows	45
Step/Actions	45
Transition	46
Jump	47
Global SFC properties	47
• Other issues	48
Changing SFC program name	48
Library warnings	48
Automatic save	49
• Printing	48
Size adjustment	49
Page setup	49
Print	50

28 April, 2018

Editor version: SFC 0.4.2.1

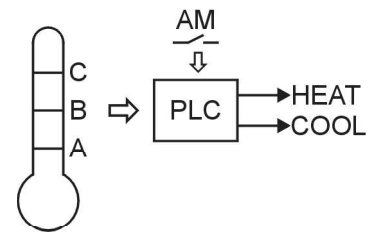
Introduction

CPDev environment supports programming in graphic SFC language (*Sequential Function Chart*) by means of *SFC Editor* module (external component). Users can create programs and function blocks, and trace values of variables during simulation. In this instruction creating typical programs in SFC language is described in details by means of three examples. First of them, temperature control in a *greenhouse*, includes steps, transitions, jumps, and sequence selections. Transition conditions are simple, with Boolean variables only. Conditions involving time are needed in the second example, *reverse switching*. Third example, level control in two *tanks*, requires concurrent simultaneous sequences. The three examples are addressed especially to the users without experience in SFC language, but familiar with basics of using CPD environment. Final part of the instruction generalizes SFC programming properties and describes some issues not covered in the examples.

Greenhouse

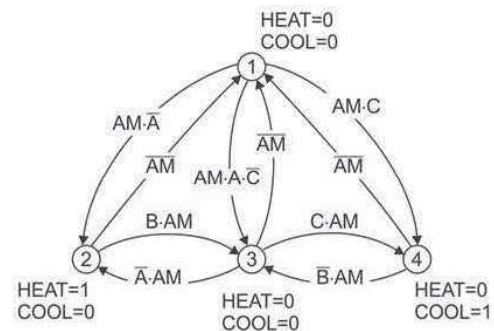
Control problem

- If AM switch is on (Auto), PLC keeps temperature in a greenhouse between low A and high C, preferably near middle B. If the temperature drops below A, heater HEAT is turned on. It is turned off when the temperature reaches B. Fan COOL is turned on when the temperature exceeds C and turned off when it drops to B. When AM switch is off (Manual), PLC sets HEAT and COOL to zero. Heater and fan are then controlled by other means.



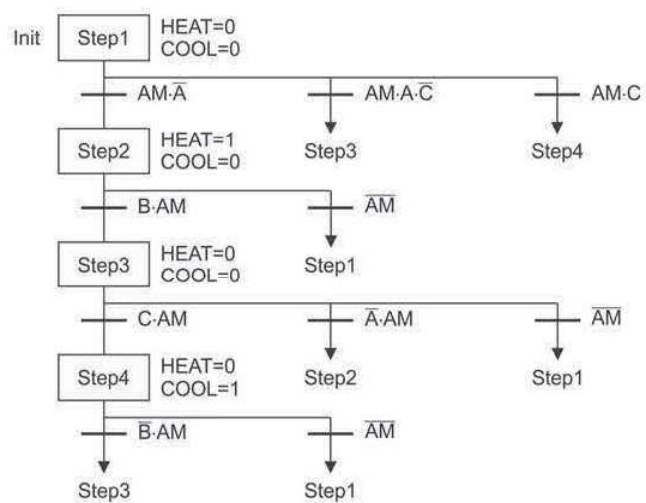
State diagram

- Simple sequential controls are conveniently represented by state diagrams. Diagram for the greenhouse consists of four states, transitions between them with appropriate conditions, and actions executed at particular states.



SFC diagram prototype

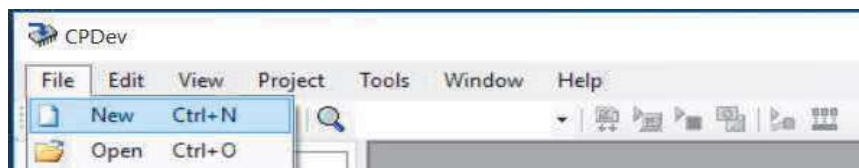
- State diagram can be directly transformed into prototype of SFC diagram with steps replacing states and bars instead of transition arrows.
- Second and the other transitions coming out of particular state are replaced by horizontal branch with jumps to relevant steps. Such branch is called *sequence selection*.
- First step is usually called *Init*.



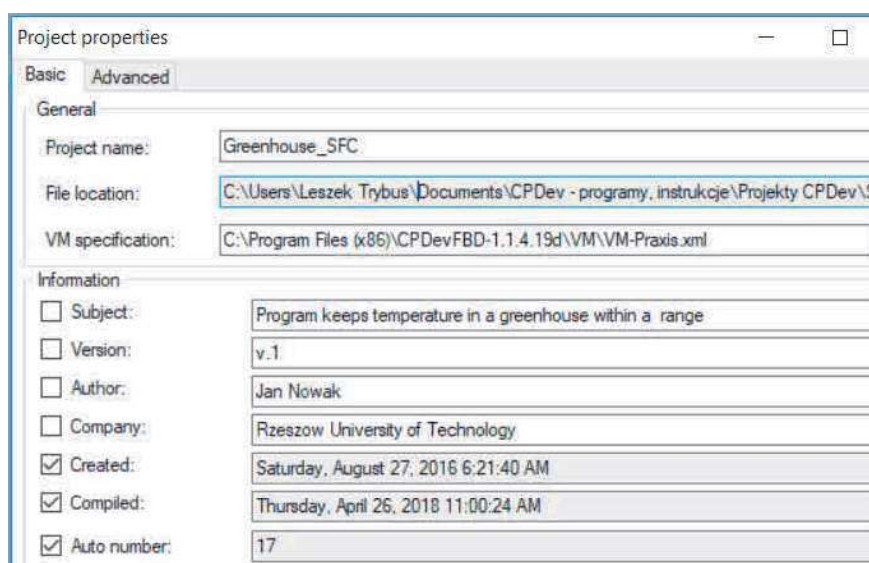
New program

Project

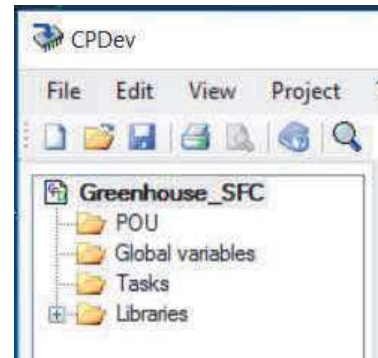
- New project is defined by selecting *File* → *New* in CPDev main menu or clicking *New* icon in the toolbar.



- Enter *Project name*, here *Greenhouse_SFC*, and eventually other data in *Project properties* window.

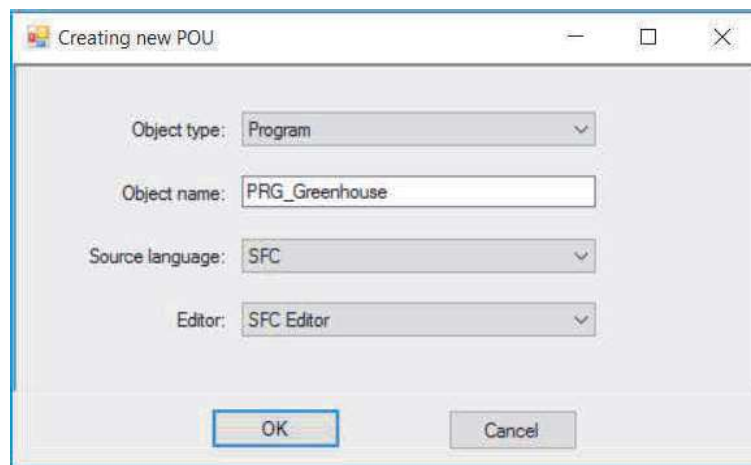


- File location, compilation date and number will be filled in automatically while saving the project.
VM specification indicates path to virtual machine description file.
- Left part of CPDev window presents initial tree of the Greenhouse_SFC project.



Program name and language

- Program can be added to the project in two ways:
 - from project tree:
 - select project name, i.e. Greenhouse_SFC
 - from context menu, select *Add Item* → *Program* or select *POU* and from its context menu choose *Add* → *Program*
 - from CPDev main menu:
 - select *Project* → *Item* → *Add*
 - select *Program* in *Adding new element* window.
- Enter *Object name*, here PRG_Greenhouse, and choose SFC language (default *SFC Editor* appears).



- In this window the user can also select function block as object type. Functions are not created in SFC language but can be invoked from step actions.
- Project tree involves now PRG_Greenhouse program.

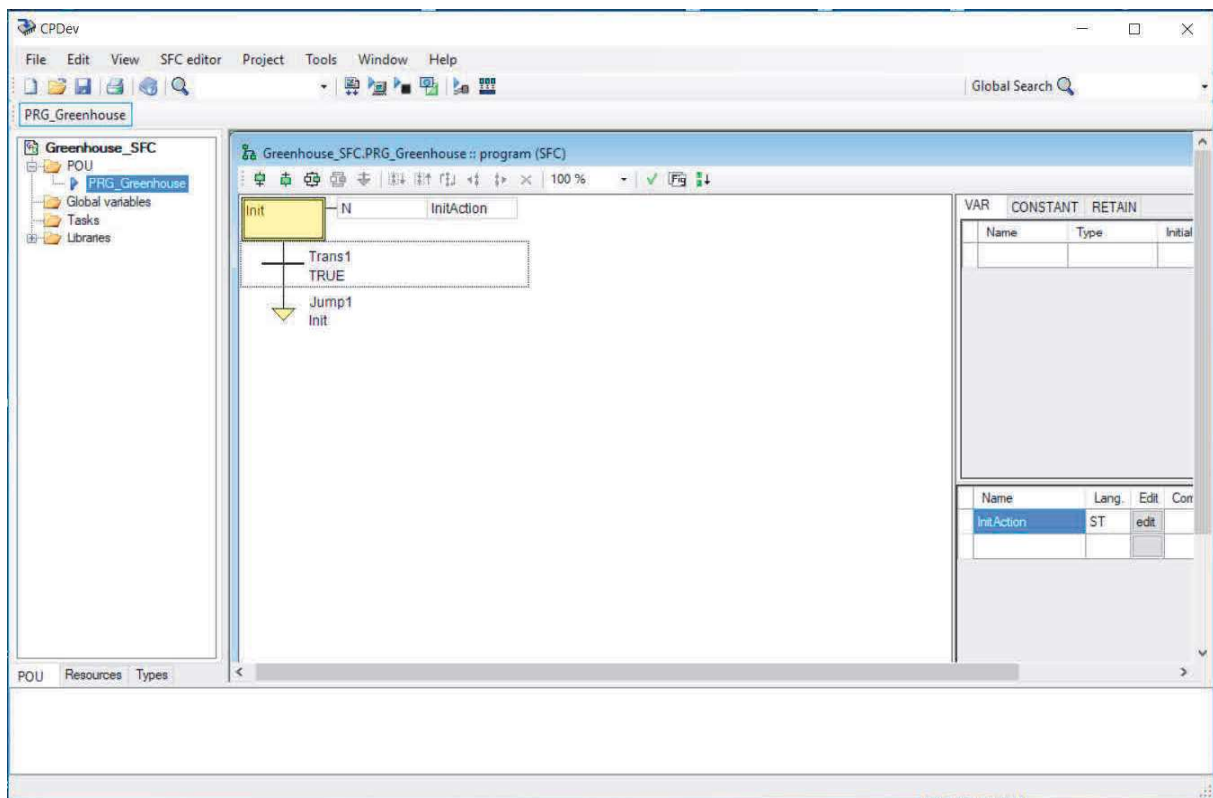


SFC graphic editor

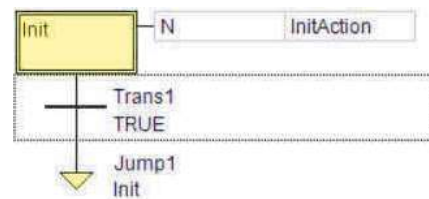
Main window

- The window involves:
 - drawing board with toolbar

- list of local variables (upper-right)
- list of actions (lower-right).



- Default diagram presented in the drawing board consists of step *Init* with *InitAction*, transition *Trans1* with TRUE condition and jump *Jump1* back to *Init*. *N* qualifier adjacent to *Init* means that *InitAction* is executed when the step is active. Frame surrounding *Trans1* indicates active element (selected).
- List of local variables is empty. Local variables, including instances of function blocks, may be needed in complicated projects (see *SFC editor summary* section).
- Program assigned to *InitAction* is initially empty what may be checked by clicking *edit* or double-clicking *InitAction*.



VAR	CONSTANT	RETAIN		
Name	Type	Initial	Comment	

Name	Lang.	Edit	Comment
InitAction	ST	edit	

Remark. When the diagram becomes sufficiently wide, a grey line appearing on the right indicates the limit of the printed page. Grey lines partition total work area into a few pages (see *Printing*).

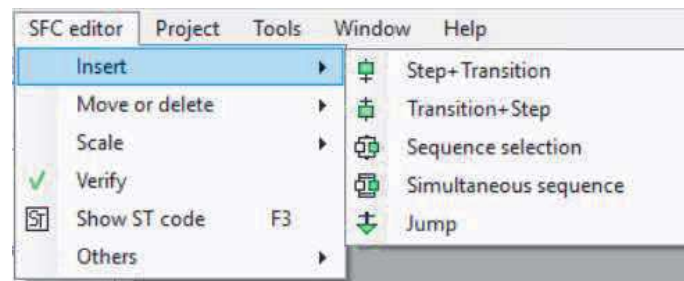
Editor options

- The following options are available in *SFC editor* and toolbar menus:



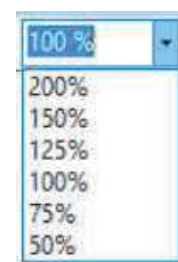
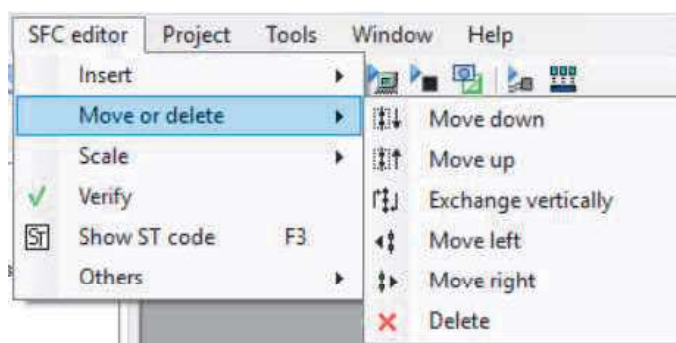
Insert – basic operations with elements added to the diagram


- Step + Transition* – new step with following transition
- Transition + Step* – new transition with following step
- Sequence selection* – new branch with alternative sequence of steps and transitions
- Simultaneous sequence* – new branch with simultaneous sequence (concurrent) of steps and transitions
- Jump* – new element which moves execution token to another step





Move or delete – basic operations for editing the diagram

- Move down* – moves selected steps and transitions down in the diagram (numbers of steps and transitions must be equal)
- Move up* – moves selected steps and transitions up in the diagram (numbers of steps and transitions must be equal)
- Exchange vertically* – exchanges upper and lower elements adjacent to selected group of steps or transitions
- Move to the left* – moves selected branches of *Sequence selection* or *Simultaneous sequence* to the left
- Move to the right* – moves selected branches of *Sequence selection* or *Simultaneous sequence* to the right
- Delete* – deletes selected steps and transitions (numbers of steps and transitions must be equal)



- Scale** – chooses scale ratio for the drawing board
- Verify**  – verifies diagram completeness (names of steps, actions, variables, etc.)
- Show ST code** – generates the code after pressing F3 (after *Build*, *Verify* invoked internally)

Others

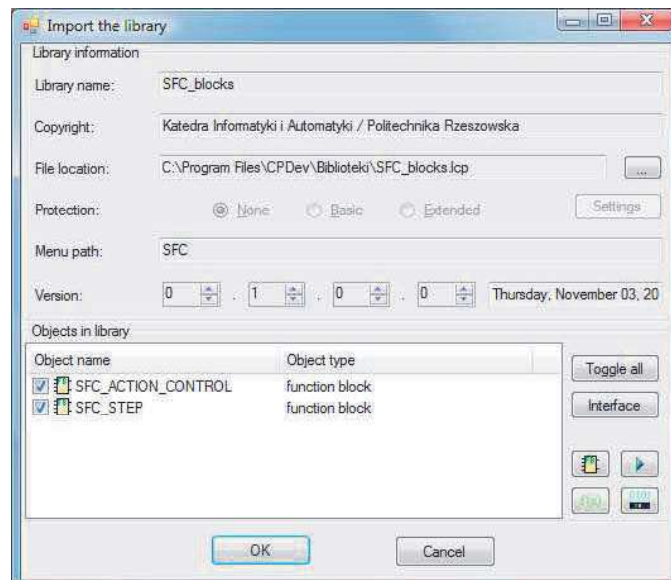
-  **Save figure** – saves figure of the diagram in PNG file
-  **Renumber** – rennumbers steps, transitions, and jumps from left-to-right and top-to-bottom.



Remark. Numbers of steps, transitions, and jumps are provided automatically by the editor. It may happen that some elements with lower numbers are at the bottom of the diagram and vice versa. The optional *renumber* function creates natural order of the numbering. Hence the renumbered diagram, especially complicated, is easier to examine.

SFC library

- SFC_blocks* library is required to execute programs in SFC language. The library is imported automatically while opening SFC editor window. However, if removed accidentally and not present in *Libraries* of the project tree, it must be imported from corresponding file in CPDev installation folder by selecting *Project* → *Import* → *Library*.



- The library consists of two function blocks which execute actions and handle steps.

Step monitoring

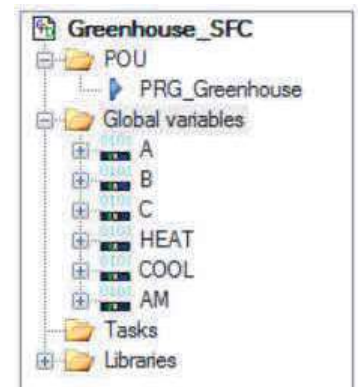
- State of a step can be monitored by two system variables:

- *state_name.X* of BOOL type which is TRUE when the step is active (FALSE otherwise)
- *state_name.T* of TIME type indicates time elapsed since activation of the step.
- The *step_name.X,T* variables may be used in transition conditions, actions and testing (simulation).

Development steps

Global variables

- Global variables are usually declared at the beginning of creating the project (see basic CPDev instruction).
- Greenhouse_SFC project involves A, B, C, HEAT, COOL and AM variables of BOOL type.



Initial SFC diagram

Complete SFC diagram with steps, transition conditions as TRUE, no action except empty *IntiAction*, and natural numbering of elements is called initial. The diagram is constructed as follows:

- Place all elements on the drawing board, preferably in the following order:
 - main vertical branch with *step + transition* pairs
 - parallel vertical branches as *simultaneous sequences* (concurrent)
 - horizontal *selection sequence* branches with transitions
 - *jumps* to appropriate steps in selection sequence branches.
- Verify structure of the diagram.
- Renumber the elements.

Transitions and actions

- Write conditions for transitions.
- Prepare list of actions and write corresponding programs for the actions.
- Bind actions to steps.
- Verify the target diagram.



If names of steps, transitions, actions, and local variables are consistent, the diagram is translated into ST language.

Task. Compilation. Simulation

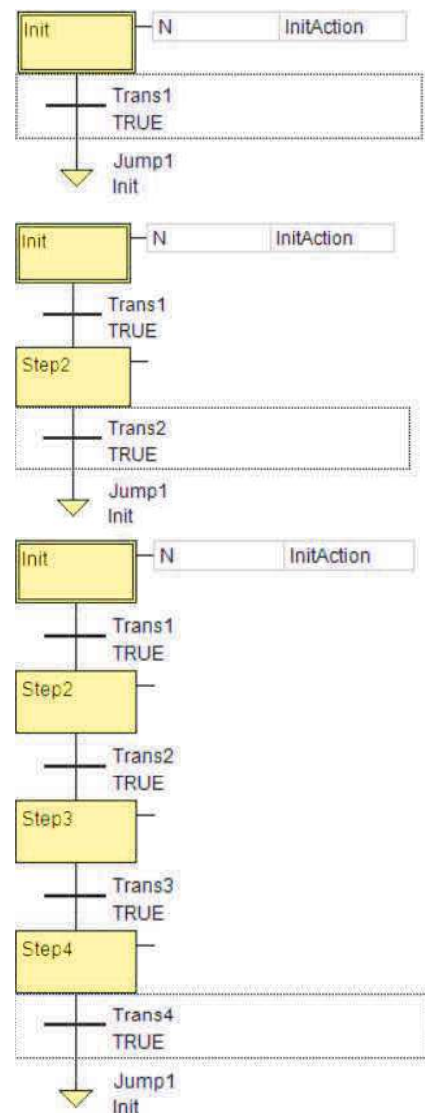
- Declare task, select cycle and assign program (programs) to the task.
- During compilation the whole project (not only the diagram) is translated into ST language. If the ST translation does not have errors, it is compiled into VMASM code of virtual machine.
- Online operating mode and CPSim simulator provide tracing of variable values during execution.

Initial SFC diagram


Main branch

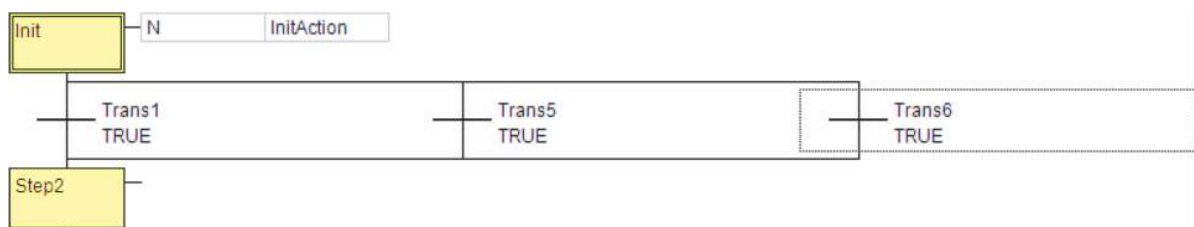
- Select *Trans1* in the default diagram (if not selected).
- Click *step + transition* icon  or choose such option in SFC editor menu to add *Step2* with *Trans2*.
Remark. No action is bound to *Step2* for the time being.
- Click the icon  twice more. The main vertical branch is ready.

Remark. Maximizing the editor window keeps the toolbar visible (scrollbars move inside).



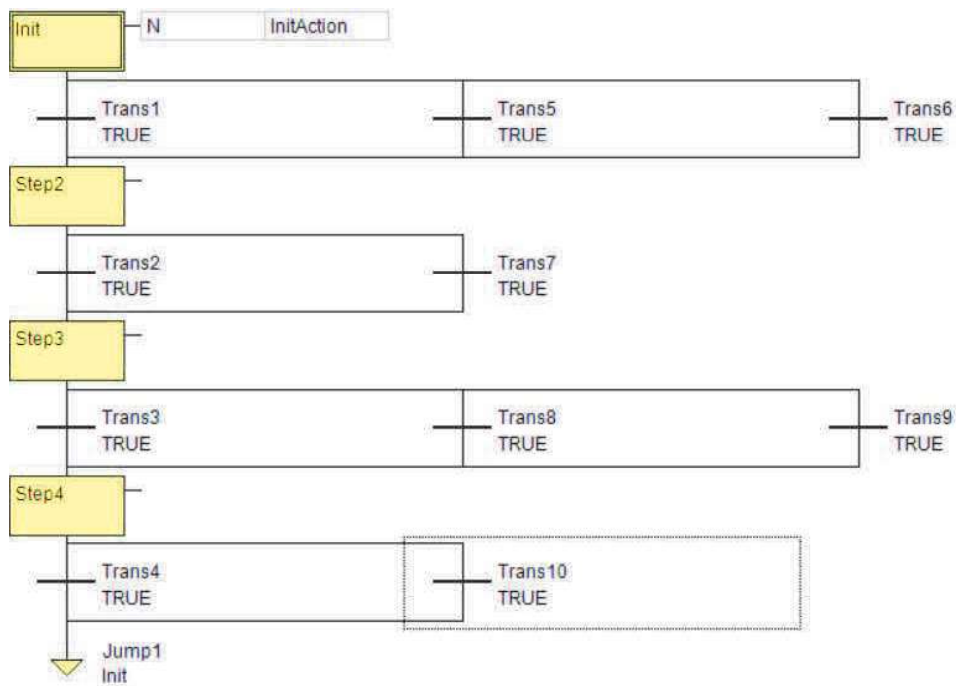
Sequence selection

- Select *Trans1*.
- Click *sequence selection* icon  or choose editor option to add *Trans5*. Click the icon once again. Horizontal branch with *Trans1*, *Trans5* and *Trans6* is created.




Remark. Grey line on the right indicates the limit of the printed page.

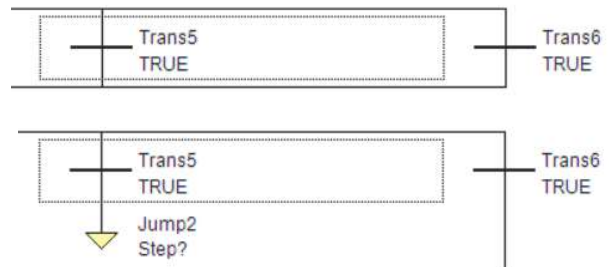
- Likewise, create sequence selection branches for *Trans2*, *Trans3* and *Trans4*.



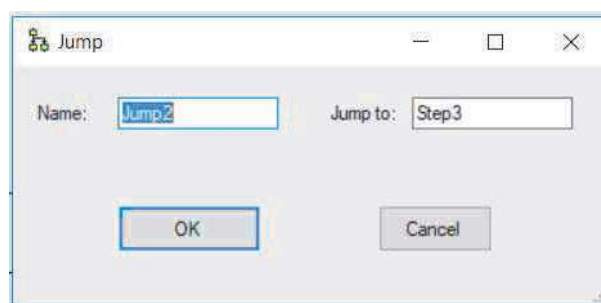
Remark. Last transition or step placed on the diagram by the editor is active (framed).

Jumps

- Select *Trans5* in the first horizontal branch.
- Click *jump* icon  or choose editor option. *Jump2* to undefined *Step?* is created.



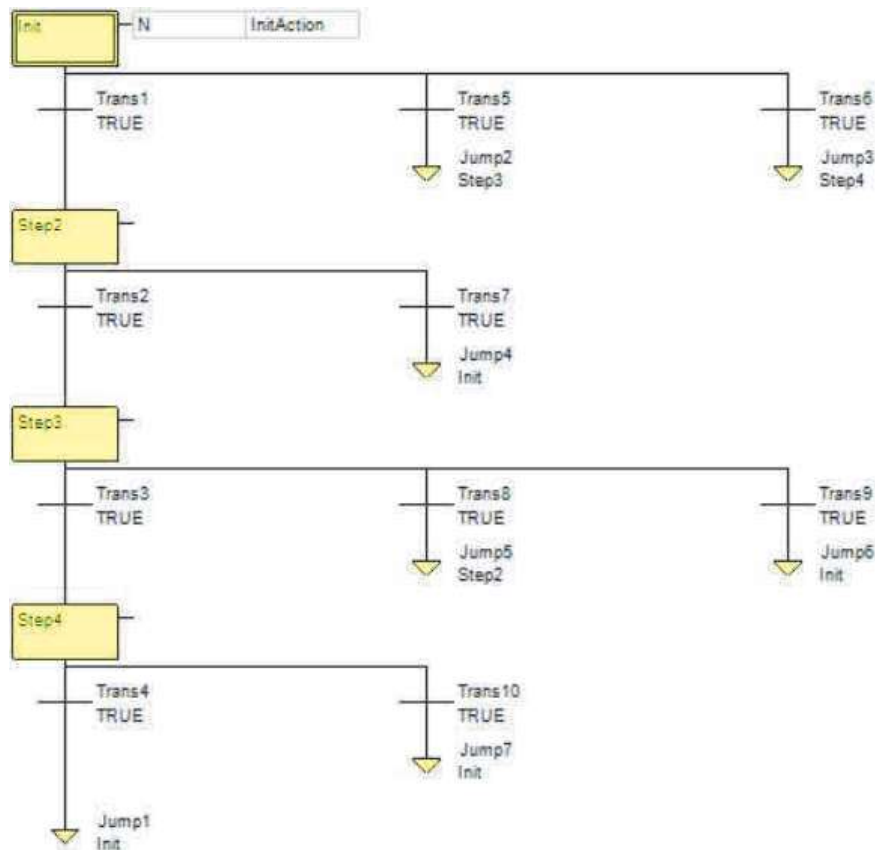
- Double-click *Jump2* to replace *Step?* by *Step3* in *Jump* properties window.



- Likewise, create *Jump3* to *Step4*.



- By providing all transitions in sequence selection branches with jumps to appropriate steps the *initial SFC diagram* is created.

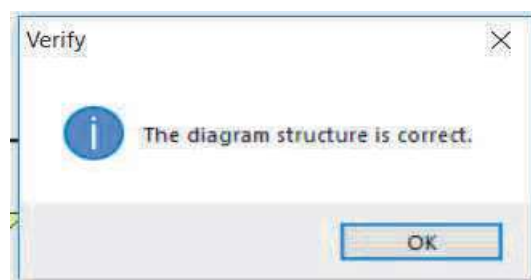


Remark. Alternatively, one may prefer to leave *Step?* temporarily for a few jumps to make corrections later. However, if it takes too long, CPDev compiler reminds by error message after some time.



Verification


- After saving the project, click *verify* icon  or choose editor option.

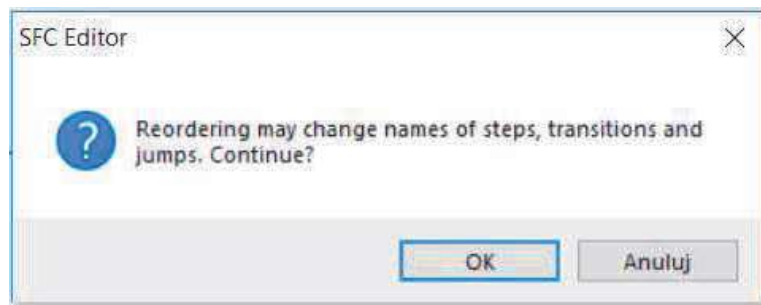


- In case of an error, a window with relevant information appears. Then make a correction (see *Diagram corrections*), save the project and verify it again.

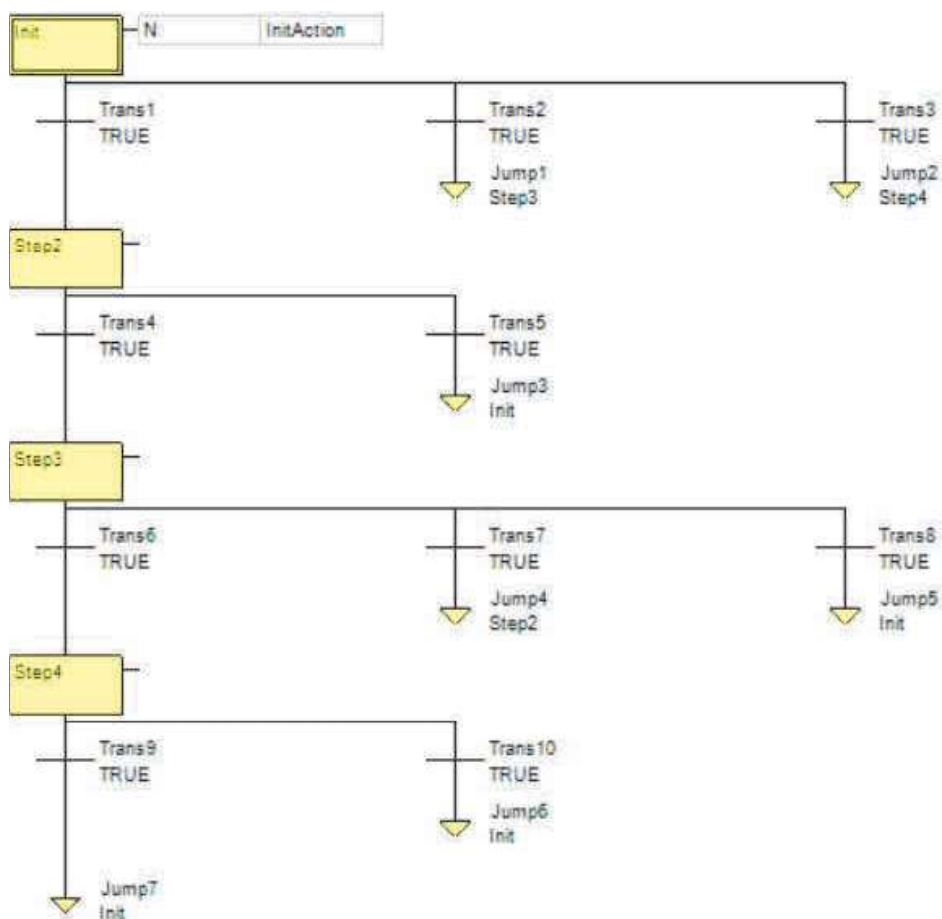
- Open diagram is automatically saved every 10 minutes into temporary directory followed by verification (see *Automatic save*). Errors are reported only.

Renumbering

- Note that currently *Jump1* is at the bottom. Click the icon  to create natural order of the numbers.



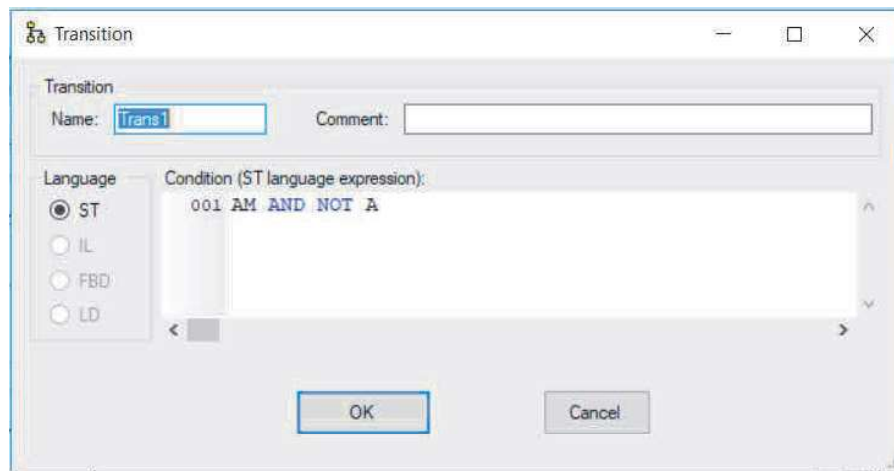
- After accepting the initial diagram is obtained.



Conditions and actions

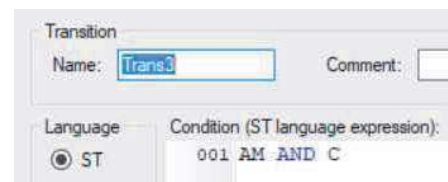
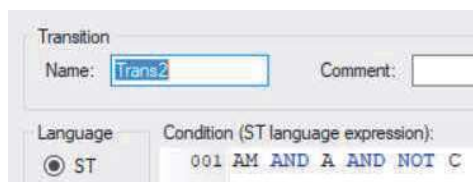
Transition conditions

- Double-click *Trans1*. Write transition condition instead of default TRUE in *Transition* properties window (see diagram prototype at the beginning).



Remark. Notice that there is no semicolon at the end of expression condition. Condition expressions must return TRUE or FALSE.

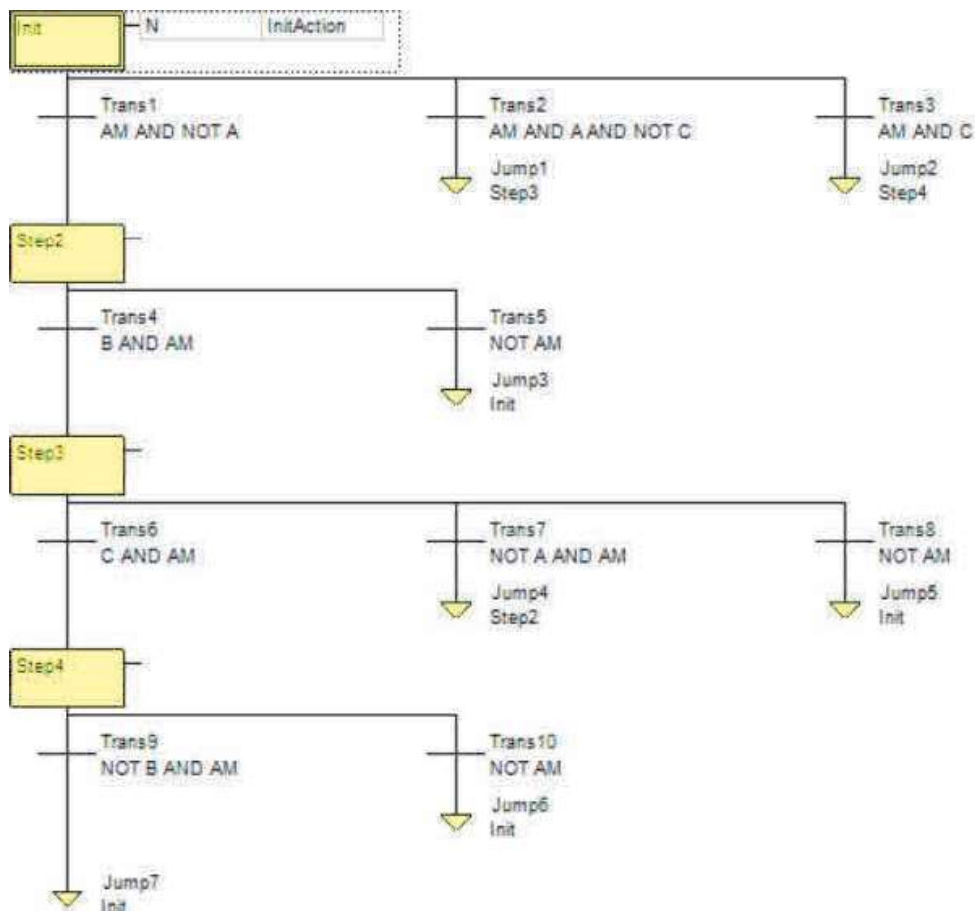
- Likewise, write transition conditions for *Trans2*, *Trans3*, etc.



Trans4: B AND AM
Trans5: NOT AM C
Trans6: C AND AM
Trans7: NOT A AND AM

Trans8: NOT AM
Trans9: NOT B AND AM
Trans10: NOT AM

- The diagram assumes the following form.



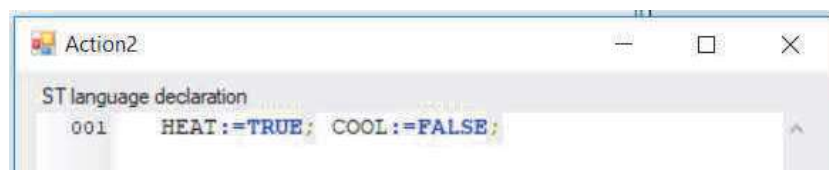
Actions

- Click edit at *InitAction* row at the lower-right window (or double-click *InitAction*) to open *InitAction* properties window and write in assignments of the output variables.

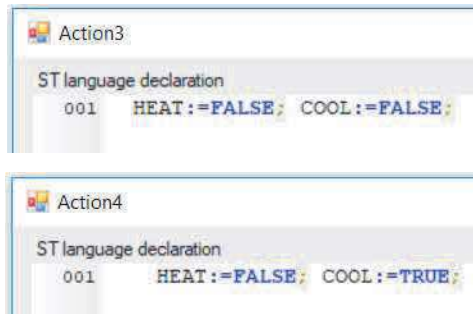
Name	Lang.	Edit	Comment
InitAction	ST	edit	



- Let *Action2,3,4* be the names of actions executed at *Step2,3,4*, respectively. Enter *Action2* in the next row of the action list and write appropriate output assignments.



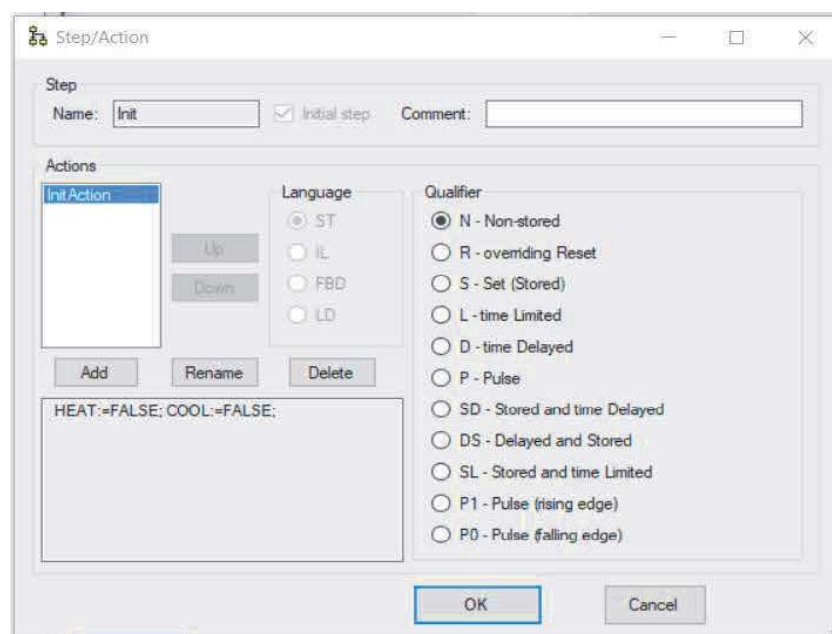
- Complete the list of actions with two remaining steps and write output assignments.



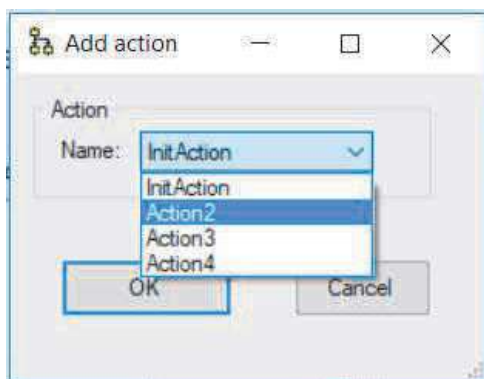
Name	Lang.	Edit	Comment
InitAction	ST	edit	
Action2	ST	edit	
Action3	ST	edit	
Action4	ST	edit	

Binding actions to steps

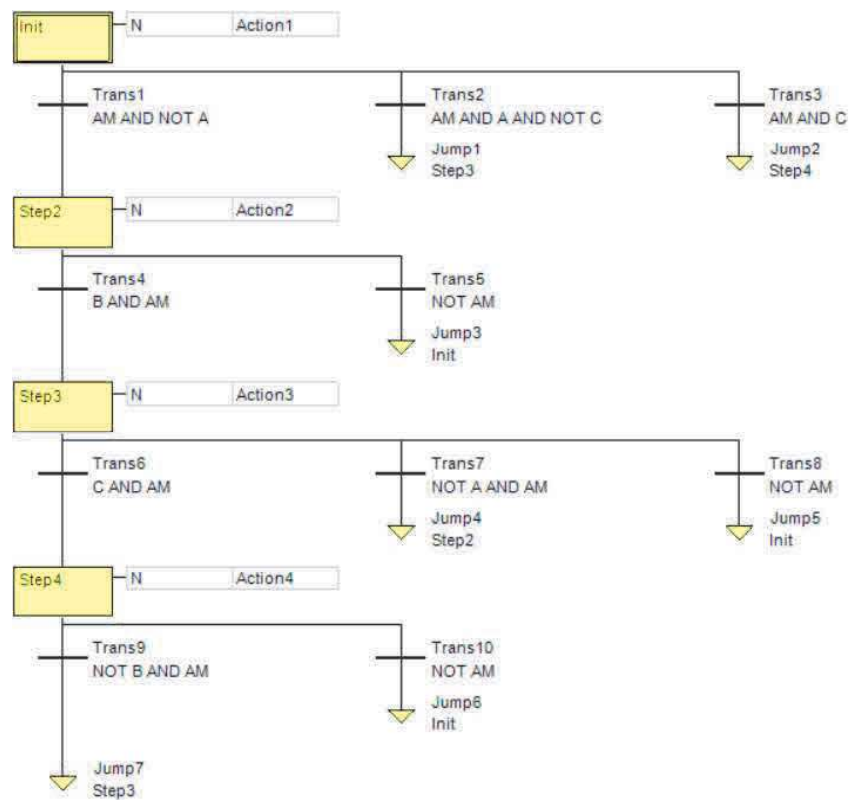
- Double-click the step *Init*. *Step/Action* properties window appears with default *InitAction* bound to *Init*. The output assignments are in the lower-left part.



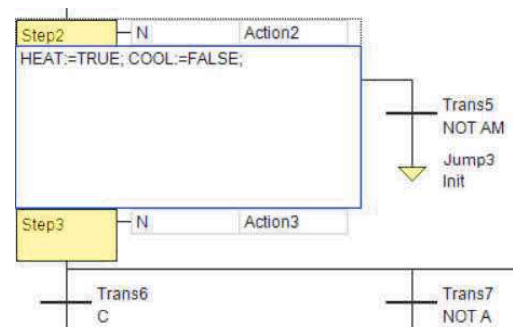
- Double-click *Step2* in the diagram and click *Add* in *Step/Action* window. Select *Action2* in appearing *Add action* window to bind *Action2* to *Step2*. Leave default qualifier *N*, so *Action2* is executed only when *Step2* is active. Similarly for *Step3* and *Step4*.




- Target SFC diagram is now completed.



- Code of the action bound to particular step can be seen in the diagram by left-click of the action name. This helps to examine complicated diagrams.



Verification and ST code

- Verify the diagram again by clicking .
- After successful verification the SFC diagram is translated into ST code. If needed, the code may be displayed by pressing F3 or by choosing corresponding option in SFC editor.

```

001 PROGRAM PRG_Greenhouse
002 VAR_EXTERNAL (*SAUTO*) END_VAR
003 VAR
004 (* ----- System variables ----- *)
005 Init : SFC_STEP;
006 Step2 : SFC_STEP;
007 Step3 : SFC_STEP;
008 Step4 : SFC_STEP;
009 _Init (*$HIDDENONLINE*) : BOOL := TRUE;
010 _Step2 (*$HIDDENONLINE*) : BOOL;
011 _Step3 (*$HIDDENONLINE*) : BOOL;
012 _Step4 (*$HIDDENONLINE*) : BOOL;
013 _ACTION_T (*$HIDDENONLINE*) : TIME;
014 Action1 : SFC_ACTION_CONTROL;
015 Action2 : SFC_ACTION_CONTROL;
016 Action3 : SFC_ACTION_CONTROL;
017 Action4 : SFC_ACTION_CONTROL;
018 _ActionError : BOOL;
019 (* ----- User variables ----- *)
020 END_VAR
021
022 Init(IN:=_Init);
023 Step2(IN:=_Step2);
024 Step3(IN:=_Step3);
025 Step4(IN:=_Step4);
026
027 Action1(N:=Init.X);
028 IF Action1.A THEN
029 (* /----- Action1 (begin) ----- \ *)
030 HEAT:=FALSE; COOL:=FALSE;
031 (* \----- Action1 (end) ----- / *)
032 END_IF
033 _ActionError := _ActionError OR Action1.ERR;
034 Action2(N:=Step2.X);
035 IF Action2.A THEN
036 (* /----- Action2 (begin) ----- \ *)
037 HEAT:=TRUE; COOL:=FALSE;
038 (* \----- Action2 (end) ----- / *)
039 END_IF
040 _ActionError := _ActionError OR Action2.ERR;
041 Action3(N:=Step3.X);
042 IF Action3.A THEN
043 (* /----- Action3 (begin) ----- \ *)
044 HEAT:=FALSE; COOL:=FALSE;
045 (* \----- Action3 (end) ----- / *)
046 END_IF
047 _ActionError := _ActionError OR Action3.ERR;
048 Action4(N:=Step4.X);
049 IF Action4.A THEN
050 (* /----- Action4 (begin) ----- \ *)
051 HEAT:=FALSE; COOL:=TRUE;
052 (* \----- Action4 (end) ----- / *)
053 END_IF
054 _ActionError := _ActionError OR Action4.ERR;
055
056 IF Init.X THEN
057 IF AM AND NOT A THEN
058 _Init := FALSE;
059 _Step2 := TRUE;
060 ELSIF AM AND A AND NOT C THEN
061 _Init := FALSE;
062 _Step3 := TRUE;
063 ELSIF AM AND C THEN
064 _Init := FALSE;
065 _Step4 := TRUE;
066 END_IF
067 END_IF
068 IF Step2.X THEN
069 IF B AND AM THEN
070 _Step2 := FALSE;
071 _Step3 := TRUE;
072 ELSIF NOT AM THEN
073 _Step2 := FALSE;
074 _Init := TRUE;
075 END_IF
076 END_IF
077 IF Step3.X THEN
078 IF C AND AM THEN
079 _Step3 := FALSE;
080 _Step4 := TRUE;
081 ELSIF NOT A AND AM THEN
082 _Step3 := FALSE;
083 _Step2 := TRUE;
084 ELSIF NOT AM THEN
085 _Step3 := FALSE;
086 _Init := TRUE;
087 END_IF
088 END_IF
089 IF Step4.X THEN
090 IF NOT B AND AM THEN
091 _Step4 := FALSE;
092 _Step3 := TRUE;
093 ELSIF NOT AM THEN
094 _Step4 := FALSE;
095 _Init := TRUE;
096 END_IF
097 END_IF
098 END_PROGRAM

```

- Local variables with (*\$HIDDENONLINE*) directives created automatically do not appear on variable value list in online mode.

Task and compilation

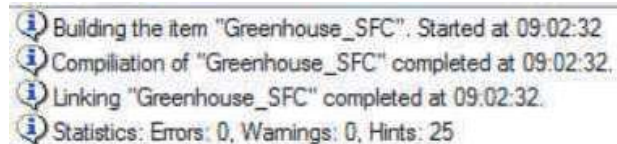
Task

- Task can be added to the project in two ways:
 - from project tree by *Tasks* → *Add task* or by *Greenhouse_SFC* (project name) → *Add item* → *Task*
 - from CPDev main menu by *Project* → *Item* → *Add* → *Task* (in *Adding new element* window).
- Enter *Task name*, here TASK_SFC, choose *Cycle interval* and *Executed programs* from *Available* ones.



Compilation – Build

- By clicking *Build* icon in the toolbar or selecting *Project → Build* in the menu the whole project is first translated into ST language and then compiled into VMAASM code for virtual machine. If errors occur, compilation is stopped. Warnings are reported, however the program can still be compiled. The project may be eventually saved before compilation (*Environment options → Editing*).
- Verification runs automatically before compilation provided that editor window is open.



Remark. Program may be compiled even without task definition, however a warning will appear.

Online mode

Tracing execution of a project is provided by CPDev online mode. The project may consist of a number of programs written in the same or different languages. Tracing is particularly convenient for graphic languages, such as LD.

Go online

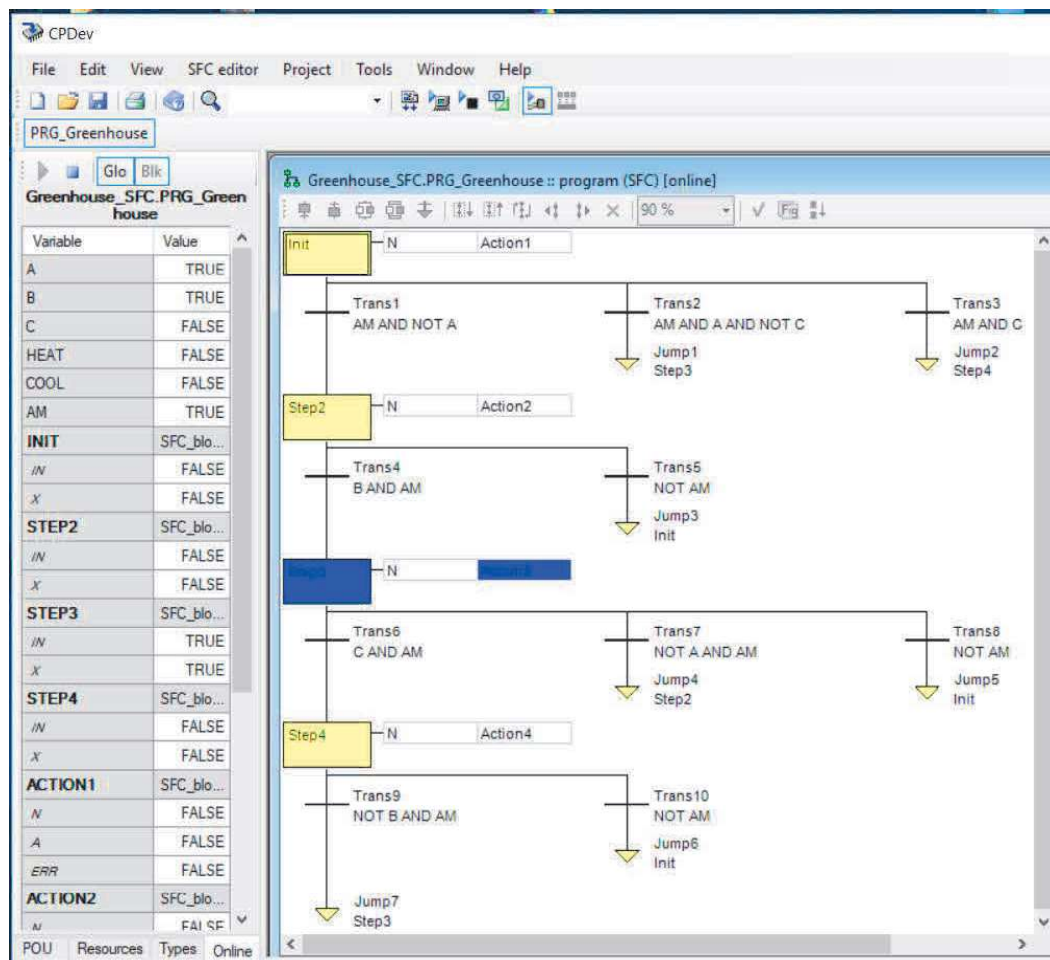
Before going into online mode, the project must be built (rebuilt after changes).

- Click *Go online* icon in the toolbar (becomes framed) or select *Project → Go online* in the menu.
- The icon indicates that *Data sources* are set to *Simulation*.



SFC diagram online

- CPDev window in online mode involves “live” SFC diagram and list of variable values on the left. Active steps and actions are shown in blue (default).



- If a few diagrams are open, the one on the top is “live”. It is indicated by [online] after name of the program.

Value list

- List with values involves variables associated with “live” diagram, so global and local variables, and function blocks with inputs/outputs. If no program is open, global variables are shown only.
- **Change value:**
 - click current value (becomes blue)
 - enter new value, accept.
- Icons above the list provide:
 - start simulation again (if stopped before)
 - stop simulation temporarily (still in online mode)
 - show/hide global variables

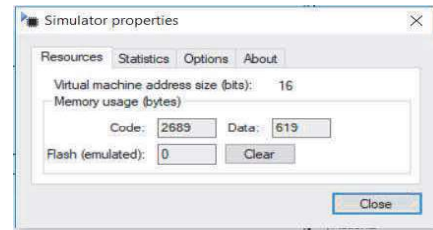
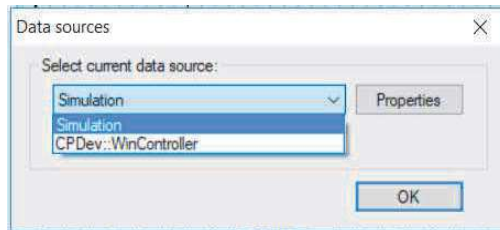
Stop freezes current values and SFC diagram. Start resumes online operation. *Blk* is inactive in SFC.
- The open list is associated with appearing *Online* tab at the bottom.

Clicking *POU* tab on the left opens project tree, for instance to choose another program for tracing. Then click *Online* again.

Variable	Value
A	TRUE
B	FALSE
C	FALSE
HEAT	FALSE
COOL	FALSE
AM	TRUE
INIT	SFC_blo...
IN	FALSE
X	FALSE
STEP2	SFC_blo...
IN	FALSE
X	FALSE
STEP3	SFC_blo...
IN	TRUE
X	TRUE
STEP4	SFC_blo...
IN	FALSE
X	FALSE

Data sources

- Clicking *Data sources* icon shows data sources for online mode, beginning with *Simulation*. Other sources, if shown, represent specific controllers for which online mode means *commissioning*. They may involve WinController, an advanced CPDev runtime for Windows.



- Some data of simulated project are shown in *Simulator properties* window.

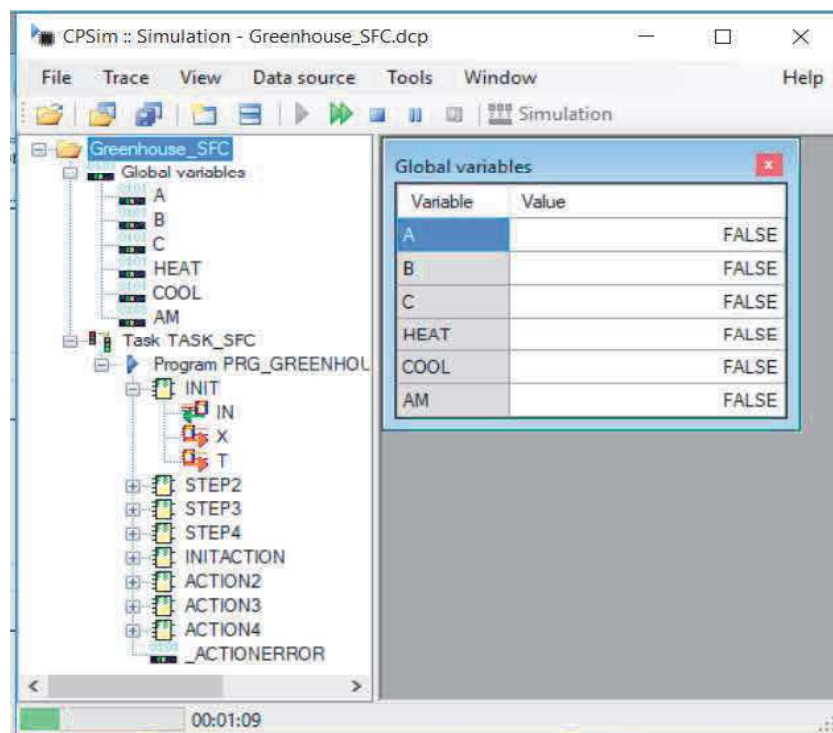
CPSim simulator

- CPSim is run by clicking *Run simulator* icon in the toolbar or by selecting *Project* → *Run simulator* (also *Tools* → *Simulator*).



Global variables

- CPSim* window appears initially with the list of global variables. Execution begins by clicking *Start trace* icon or by choosing *Trace* → *Start*.

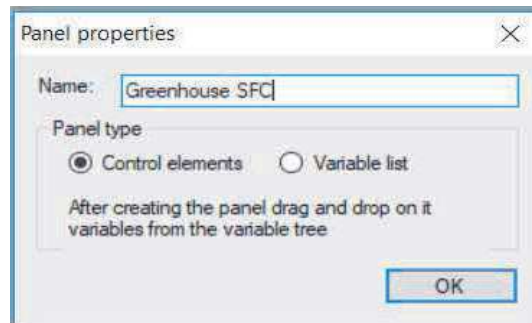


Remark. Tree of Task *TASK_SFC*, originally folded, is unfolded here. Values of boolean variables are displayed as 0 (FALSE) and 1 (TRUE).

- Values of those global variables which are inputs to the program can be changed. More on CPSim can be found in *Help → Programming instruction*.

Control panel

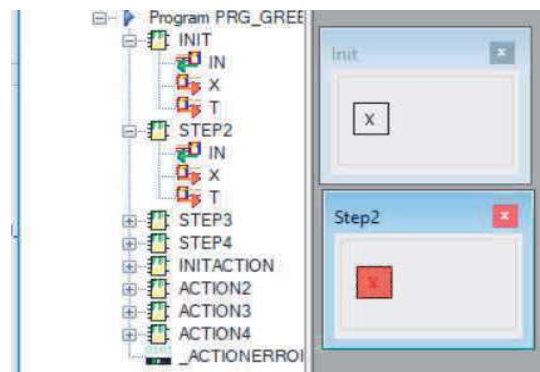
- More convenient way of testing is provided by selecting *Group panel* icon or by relevant option in *View*.



- Type name in *Panel properties* window, accept and drag required variables from simulation tree on the left into the rectangle panel (which grows accordingly).



- Test the program by pressing input buttons (BOOL) or by typing values in input cells (other types).
- Activity of particular steps during execution may be monitored by creating panels with *STEP.X* variables from *STEP* function blocks (see *Step monitoring* earlier).



- Comprehensive view involving panel with global variables and collection of all *STEP.X* panels allows to check step-by-step how the SFC diagram is executed.



Individual view

- Single variable can be traced by dragging it from the tree into view area. Individual views are used for important local variables and inputs/outputs of function blocks (STEP2.T here).

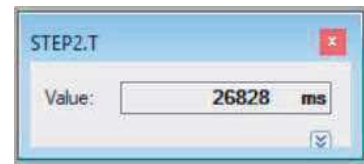
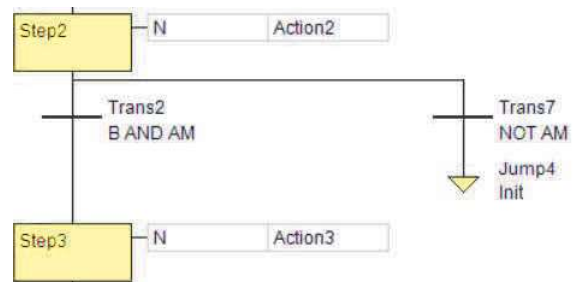


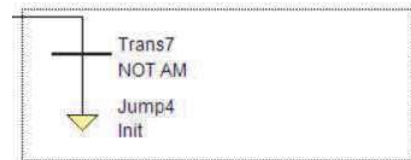
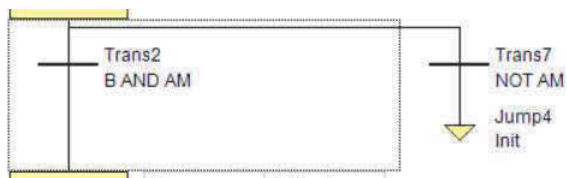
Diagram corrections

- Connections are described below using a portion of PRG_Grenhouse diagram involving two steps with sequence selection in between.

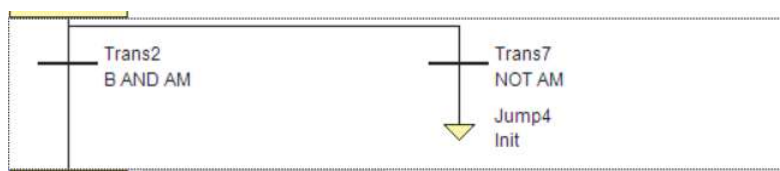
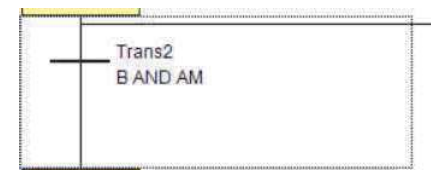


Selecting elements


- Step, transition, jump*
Click the element. Frame indicates selection. Only one element can be selected, earlier one (if any) is deactivated.
- Branch of sequence selection or simultaneous sequence*
Click horizontal line above the branch or a little above the line. Frame surrounds the branch (selected).

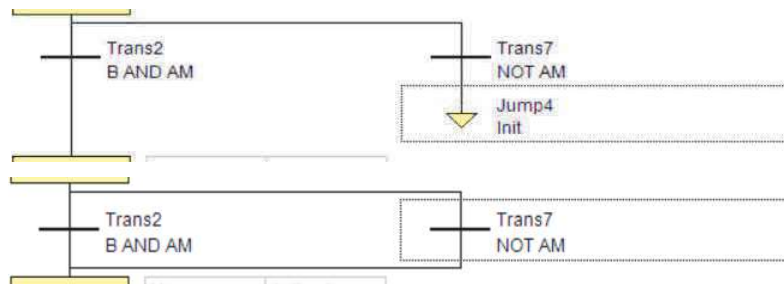


- Sequence selection or simultaneous sequence (whole)*
Select the first or last branch (as above). Click empty area inside the frame and move cursor right or left to cover all branches, keeping left-mouse button pressed (do not move the cursor too far). Release the button.



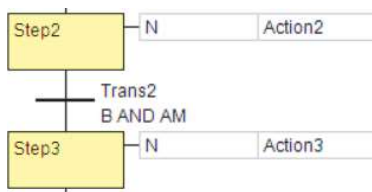
Deleting single elements

- Deleting single element is possible only when *delete* icon  is enabled (red).
- Jump*
Deleting a jump in sequence selection branch connects preceding transition to step which follows main branch.



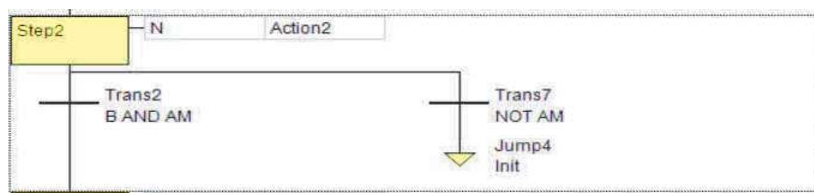
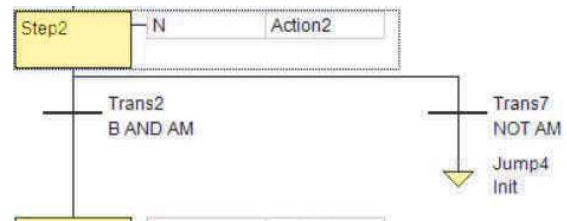
Remark. Single step or transition cannot be deleted.

- *Branch* of sequence selection or simultaneous sequence
Select a branch and click *delete* icon. The diagram is automatically corrected. The result of deleting the second branch in the diagram portion is shown below. Deleting the first branch is forbidden because at least one branch (of the simultaneous sequence) without the final jump must be left.



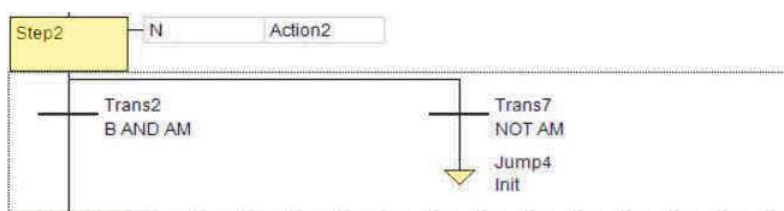
Selecting pairs

- *Step, transition or jump* as initial in the pair.
Select the initial element. Click empty area inside the frame and move cursor down or up to cover the second element, keeping left-mouse button pressed. Release the button. Frame surrounding the pair appears.



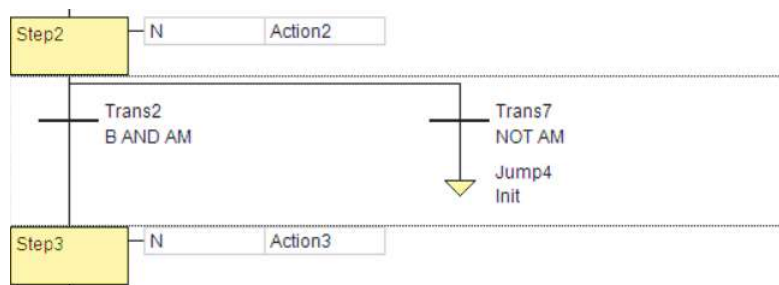
Remark. More elements can be selected in this way, however pairs are needed most often (see below).

- The same result is obtained if the whole sequence selection is selected (as below) and the cursor moved up.

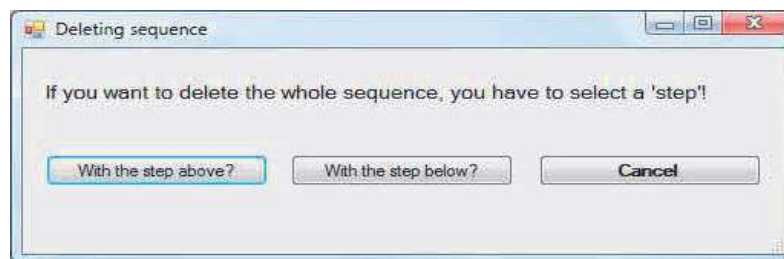


Deleting pairs

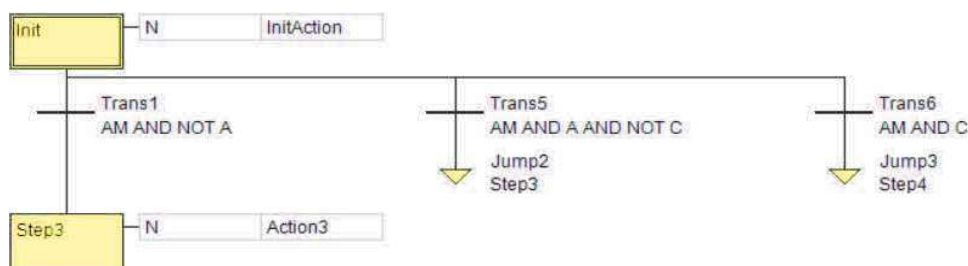
- *Step + transition or transition + step*
Such pairs, not single elements, can be deleted only. The icon *delete* becomes enabled when the pair is selected (or multiplicity of pair).
- *Sequence selection + step or step + sequence selection*
Such pairs can be deleted since sequence selection may be treated as a complex transition.
- *Simultaneous sequence + transition or transition + simultaneous sequence*
Likewise, such pair can be deleted since simultaneous sequence may be treated as a complex step.
- Pairs involving *sequence selection* or *simultaneous sequence* can also be deleted as follows:
 - Select sequence selection or simultaneous sequence. Note that *delete* icon becomes enabled.



- Click *delete* icon and choose which step, above or below, belongs to the pair.



- When *step above* is chosen (*Step2*), the pair is deleted. Initial part of PRG_Greenhouse diagram becomes as below.

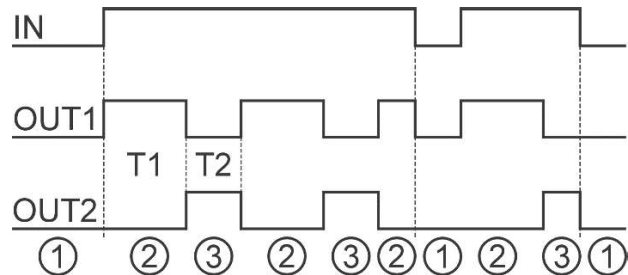


Reverse switching

Control problem

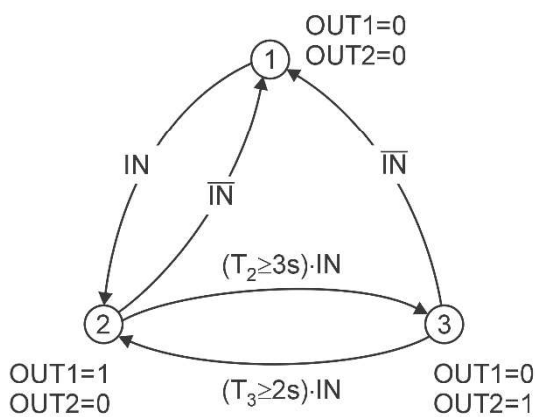
- If IN is on then OUT1 is on and OUT2 is off, or conversely. OUT1 is on for 3 seconds, OUT2 for 2 seconds. If IN is off, OUT1 and OUT2 are also off.

Remark. The example shows how to implement transitions with time conditions.

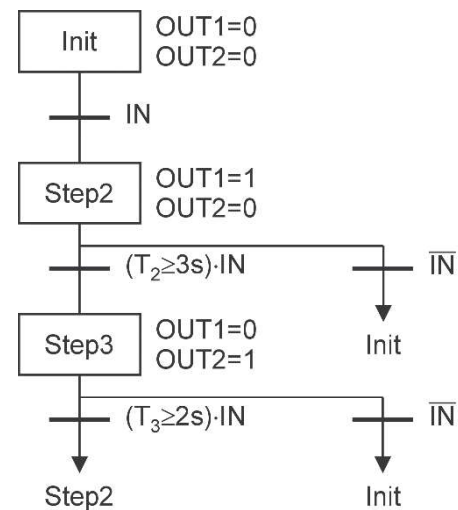


Preliminary diagrams

- State diagram



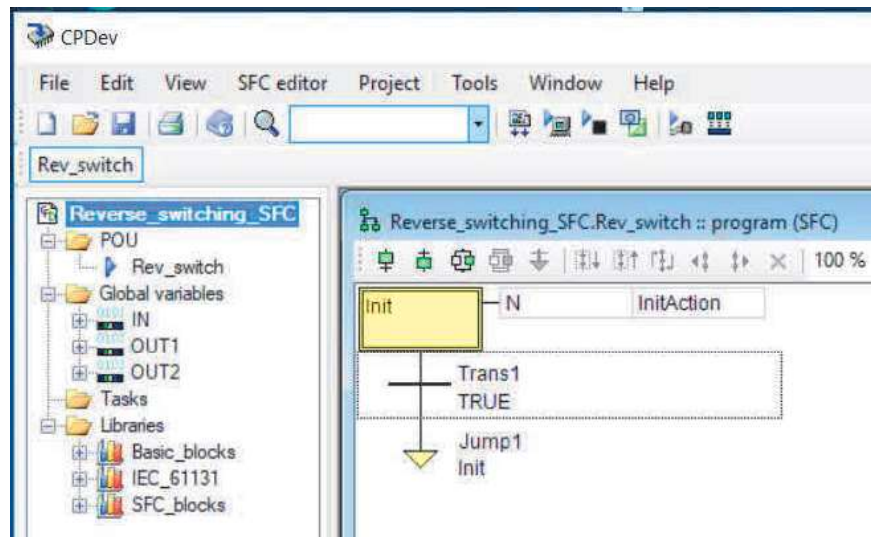
- SFC diagram prototype




- T_2 and T_3 denote execution times of corresponding states or steps. Recall from *Step monitoring* that execution times are represented by $Step2.T$ and $Step3.T$ variables in SFC editor (time elapsed since activation)

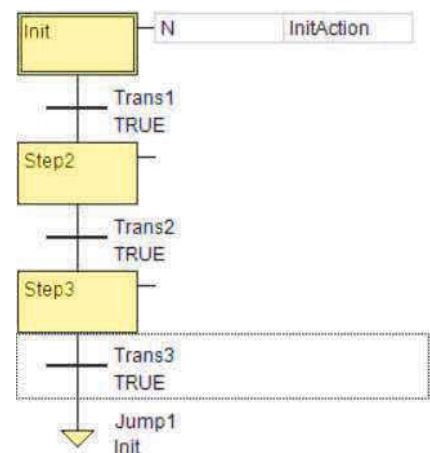
Initial SFC diagram

- Project, variables, program
Define Reverse_switching_SFC project, declare global variables IN, OUT1, OUT2 and add program Rev_switch to be written in SFC language.




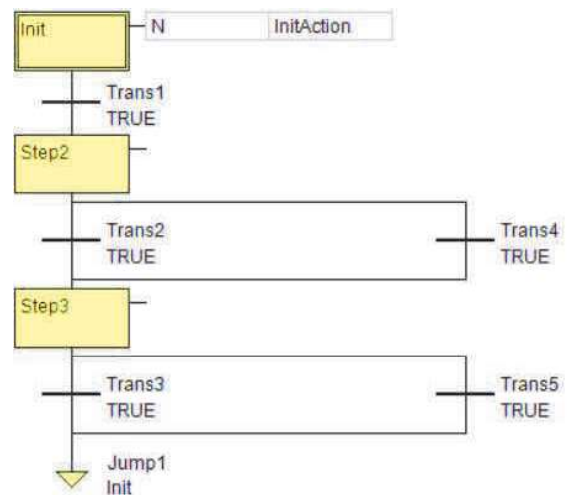
- Main branch

Select *Trans1* and click *step + transition* icon  twice (or choose editor option).






- Sequence selection branches

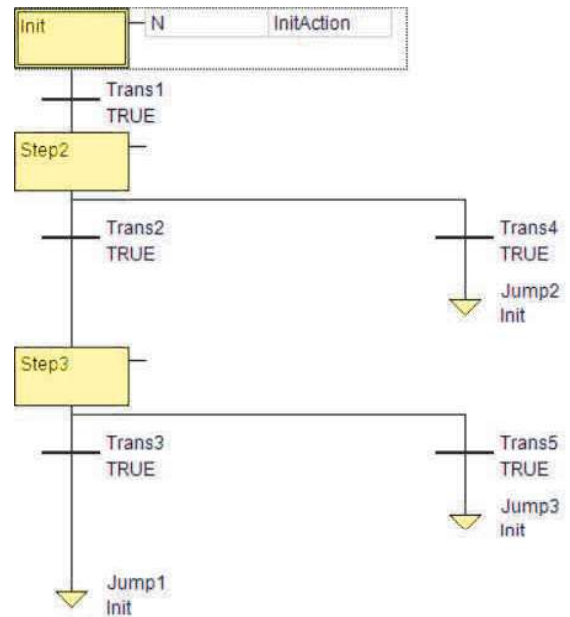
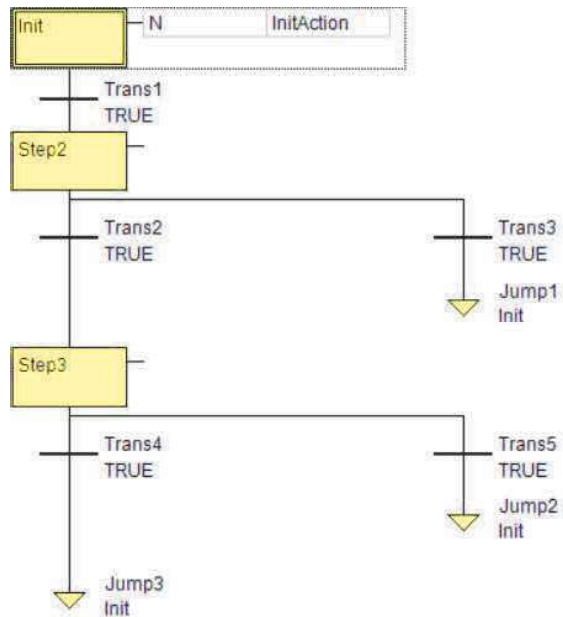
Select *Trans2* and click *sequence selection* icon . Branch with *Trans4* appears. Repeat the same for *Trans3* to get second branch with *Trans5*.



- Jumps

Select *Trans4* and click jump icon . *Jump2* with undefined *Step?* appears. Double-click *Jump2* and in appearing *Jump* properties window replace *Step?* by *Init* (see preliminary diagram). Repeat the same for *Trans5*.

- Verify resulting diagram by clicking .
- Renumber the elements by clicking .



Conditions and actions

- Transition conditions

Double-click *Trans1* and write IN as condition in *Transition* properties window. Repeat the same for *Trans2* using *Step2.T* as execution time in the condition expression (see preliminary diagram). Write conditions for the other transitions.

Transition

Name: Comment:

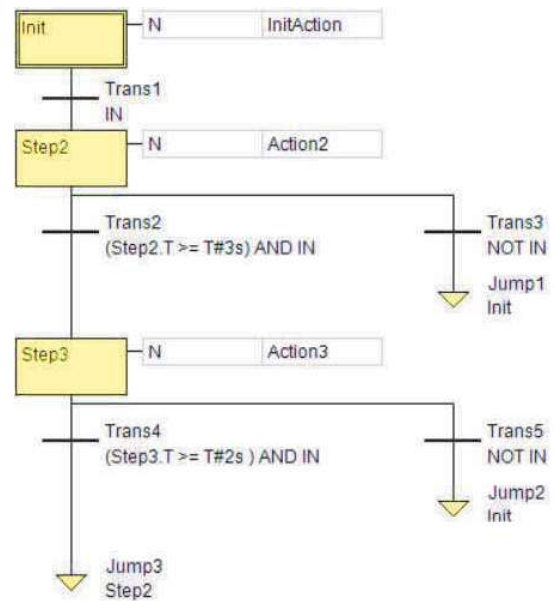
Language ☒ ST Condition (ST language expression): 001 IN

Transition

Name: Comment:

Language ☒ ST Condition (ST language expression): 001 (Step2.T >= T#3s) AND IN

Trans3: NOT IN
 Trans4: (Step3.T >= T#2s) AND IN
 Trans5: NOT IN



Actions

- Select *InitAction*, click *edit* and write output assignments in *InitAction* properties window. Enter *Action2* and *Action3*. Repeat output assignments for *Action2* and *Action3*.

Name	Lang.	Edit	Comment
InitAction	ST	edit	
Action2	ST	edit	
Action3	ST	edit	

InitAction

ST language declaration

001 OUT1:=FALSE; OUT2:=FALSE;

Action3

ST language declaration

001 OUT1:=FALSE; OUT2:=TRUE;

Action2

ST language declaration

001 OUT1:=TRUE; OUT2:=FALSE;

- Binding actions to steps
 Double-click *Step2* in the diagram to open *Step/Action* window. Press *Add* button and in appearing *Add action* window choose *Action2*. Similarly bind *Action3* to *Step3*.

Add action

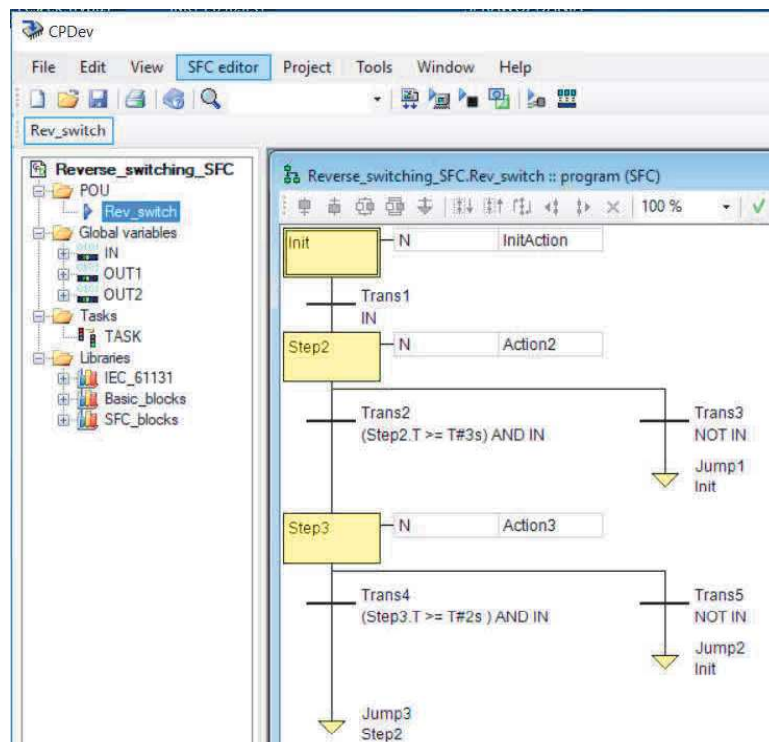
Action

Name:

OK Cancel

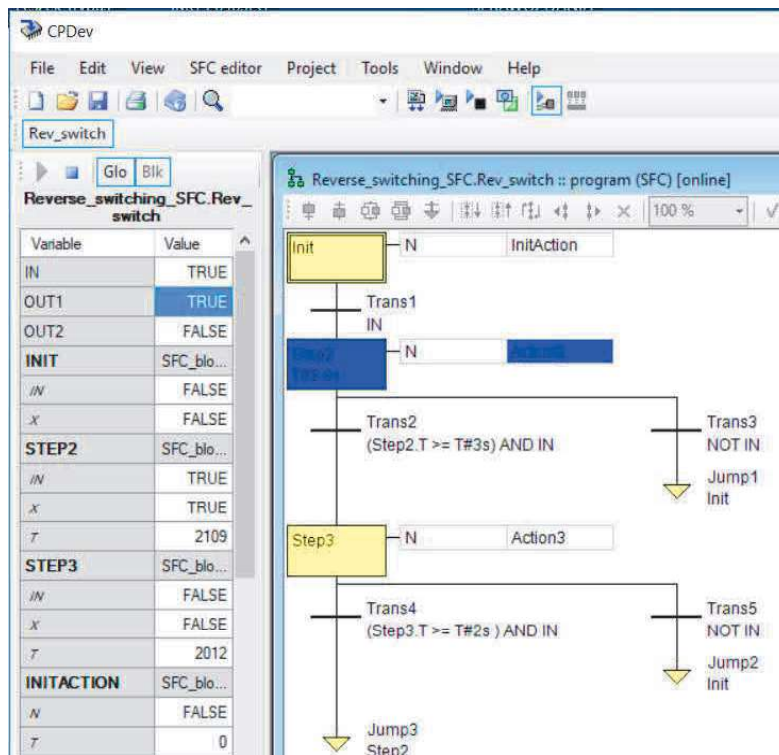
SFC diagram

- Binding actions to steps completes development of the diagram. Declare TASK_SFC with Rev_switch as executed program (see *greenhouse* project).

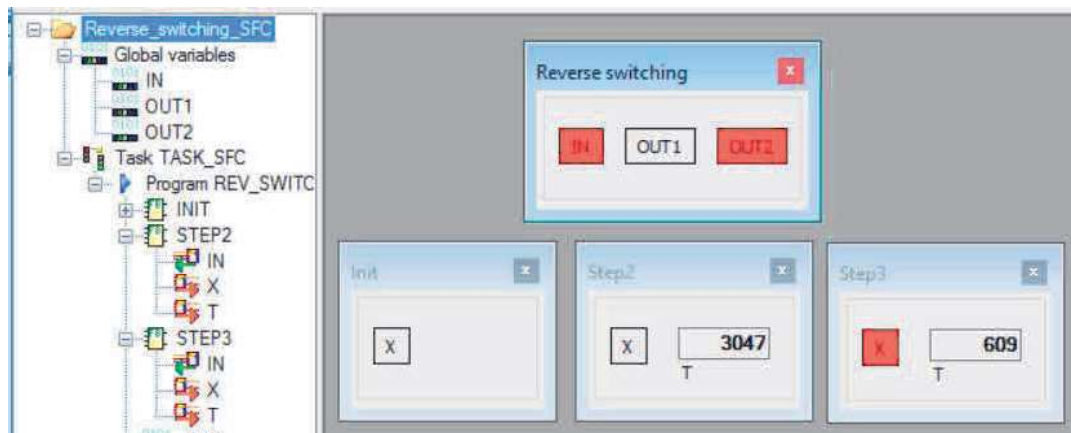


Simulation

- The diagram in the online mode for IN set to TRUE



- CPSim window shown below involves:
 - basic panel with global variables IN, OUT1, OUT2
 - three panels for monitoring SFC diagram execution with *STEP.X* and *STEP.T* variables for corresponding steps (*T* in milliseconds).

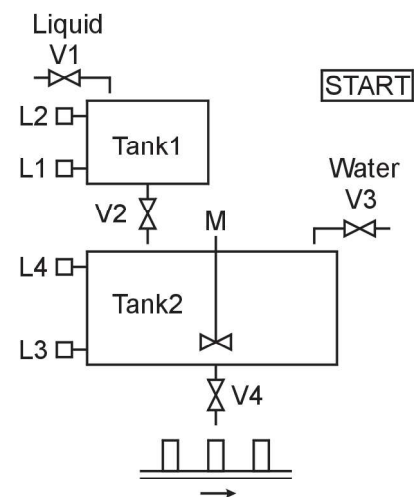


Tanks

Control problem

Condensed liquid filling upper Tank1 is mixed with water in lower Tank2. When Tank2 is full, after mixing for a while, the mixture is emptied into containers on the conveyor. Control of the two tanks proceeds as follows:

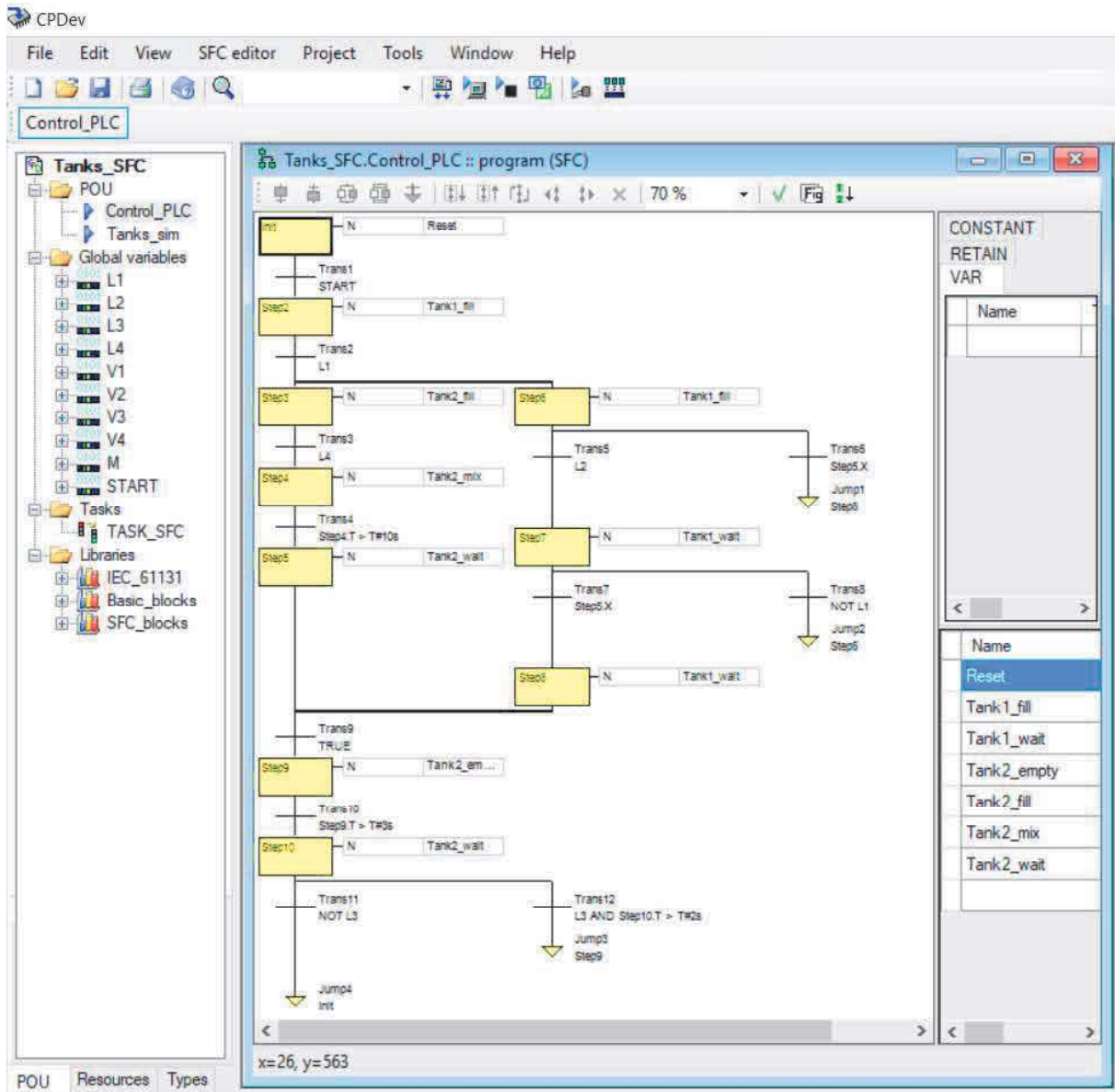
- Sequence begins after pressing self-releasing button START.
- After filling Tank1 by valve V1 at least to low level L1, the emptying valve V2 and water valve V3 are open to fill Tank2.
- Filling Tank2 by V2, V3 and Tank1 by V1 up to high level L2 proceeds simultaneously. Tank1 needs filling a few times.
- When level in Tank2 reaches L4, valves V2, V3 are closed and mixer M is switched on for 10 seconds. If Tank1 is not full yet, filling continues up to the level L2.
- After the mixing, valve V4 begins emptying Tank2 in open-for-3-seconds and closed-for-2-seconds cycles filling the containers below. Emptying ends when level drops to L3.
- New sequence begins by pressing START again.



Remark. State diagram and prototype SFC diagram are not presented here. Contrary to the previous examples, final project with target diagram is shown first, followed by similar step-by-step explanation how it is created. Particular attention is given to simultaneous sequence which handles concurrent control of the tanks. Second program written in ST language calculates volumes of liquids in the tanks, for simulation of controller-plus-tanks operation in real-time.

Tanks_SFC project

- The project involves:
 - Control_PLC program in SFC language which solves control problem
 - Tank_sim program in ST for calculation of volumes in the tanks
 - global variables L1,L2 up to START (boolean)
 - task TASK_SFC with Control_PLC and Tanks_sim executed programs.



- Target SFC diagram of Control_PLC program consists of three sections:
 - waiting for START and then filling Tank1 up to low level L1: *Init, Step2*
 - simultaneous sequence with two concurrent branches involving:
 - filling Tank2 up to L4, mixing for 10 seconds and waiting (coordination): *Step3,4,5*
 - filling Tank1 a few times and waiting till mixing is completed: *Step6,7,8*
 - emptying Tank2 in 3+2 seconds cycles: *Step9,10*.
- Conditions *Step5.X* in transitions *Trans6,7* from steps of Tank1 (right branch) verify whether mixing in Tank2 is completed.

- Simultaneous sequence section is left when “wait-for-the-other” steps *Step5* and *Step8* at the ends of the two branches are both active.
- Simultaneous sequence must be followed by transition with TRUE condition (*Trans9* in the diagram).


Initial diagram

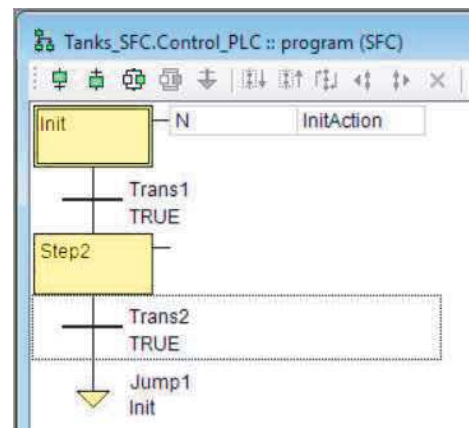
Development steps

- Initial diagram is built in the following order:
 - first section with two steps
 - simultaneous sequence section with two vertical branches, each involving three steps
 - two horizontal sequence selections coming out of the right branch of simultaneous sequence
 - third section with two steps and sequence selection between them
 - renumbering steps, transitions and jumps.



Remark. Global variables should be created at the beginning.

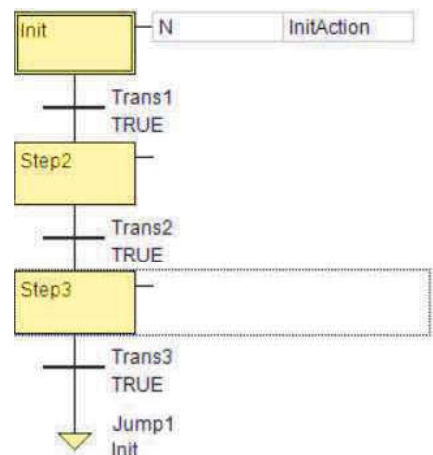
First section

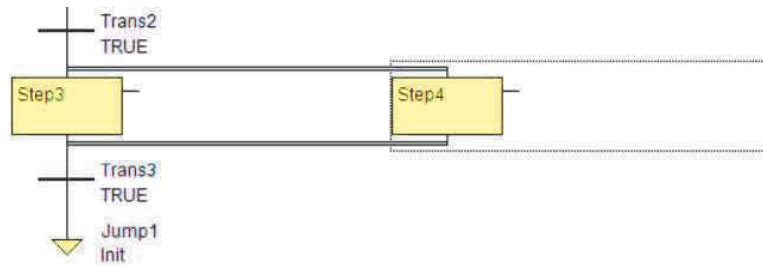
- Given default SFC diagram with *Init*, *Trans1* and *Jump1*, select *Trans1* and click *step + transition* icon  (or choose editor option)




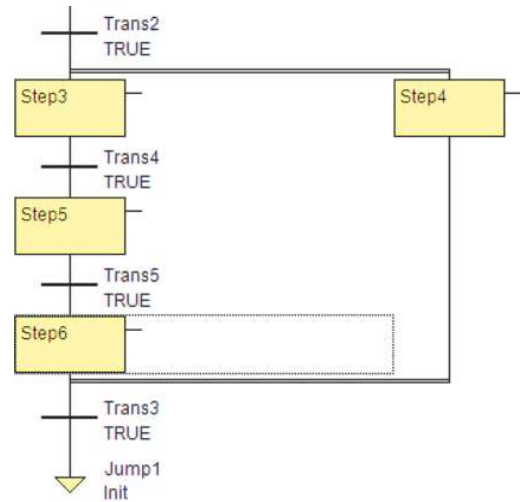
Simultaneous sequence

- Select *Trans2*, click the icon  again to get *Step3* which will be the first step in simultaneous sequence. Select *Step3*.
- With *Step3* selected, click *simultaneous sequence* icon  what creates simultaneous sequence with two concurrent steps, *Step3* and *Step4*.

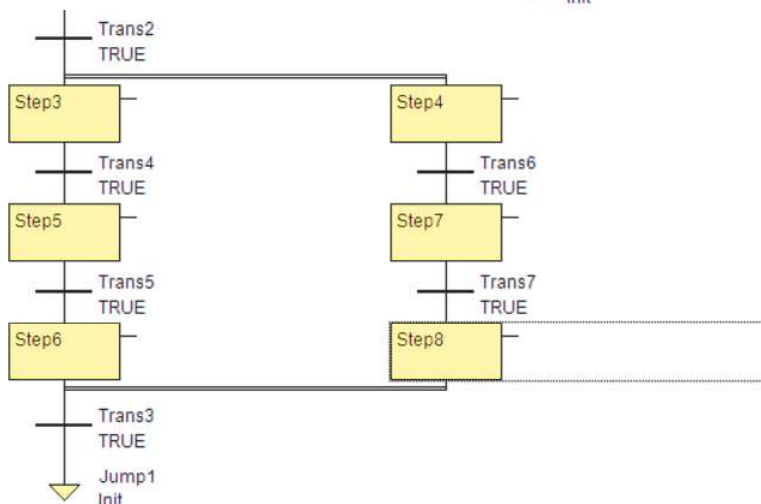





- To fill the left branch select *Step3* again and click *transition + step* icon  twice to add *Step5* and *Step6*.

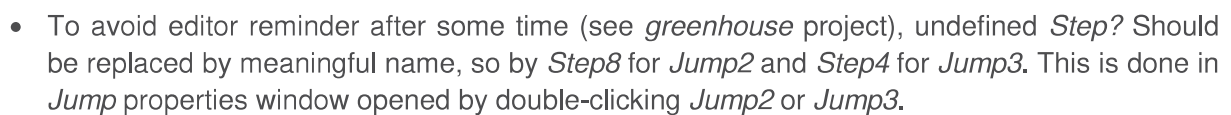


- Repeat the same for *Step4* in the right branch adding *Step7* and *Step8*.




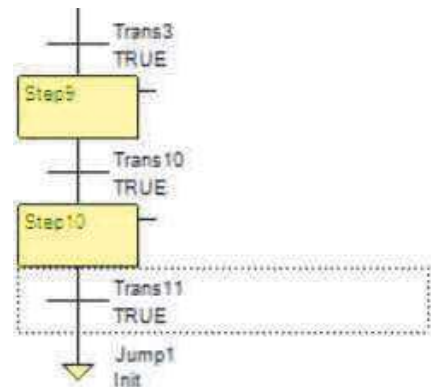
Sequence selection branches



- Select *Trans6* in the right vertical branch and click *sequence selection* icon  to add horizontal branch with *Trans8* on the left. Repeat the same for *Trans7* adding second branch with *Trans9*.

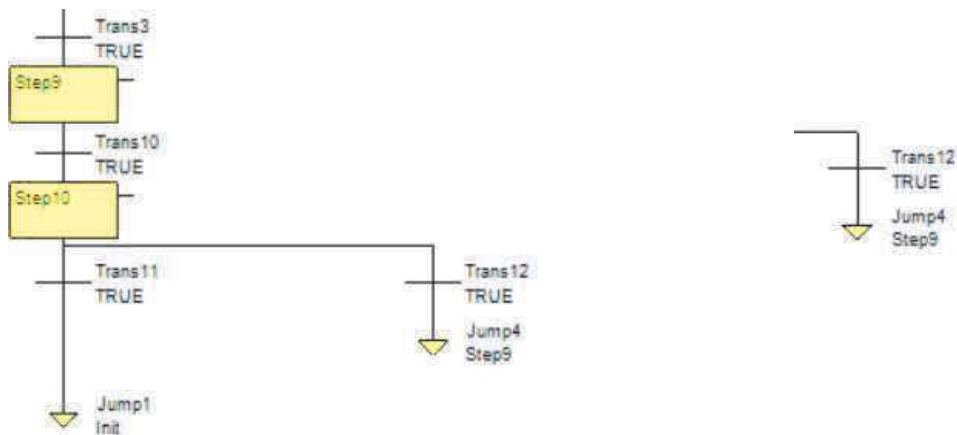


Third section

- Select *Trans3* after the simultaneous sequence and click *step + transition* icon  twice. *Step9* and *Step10* are created.

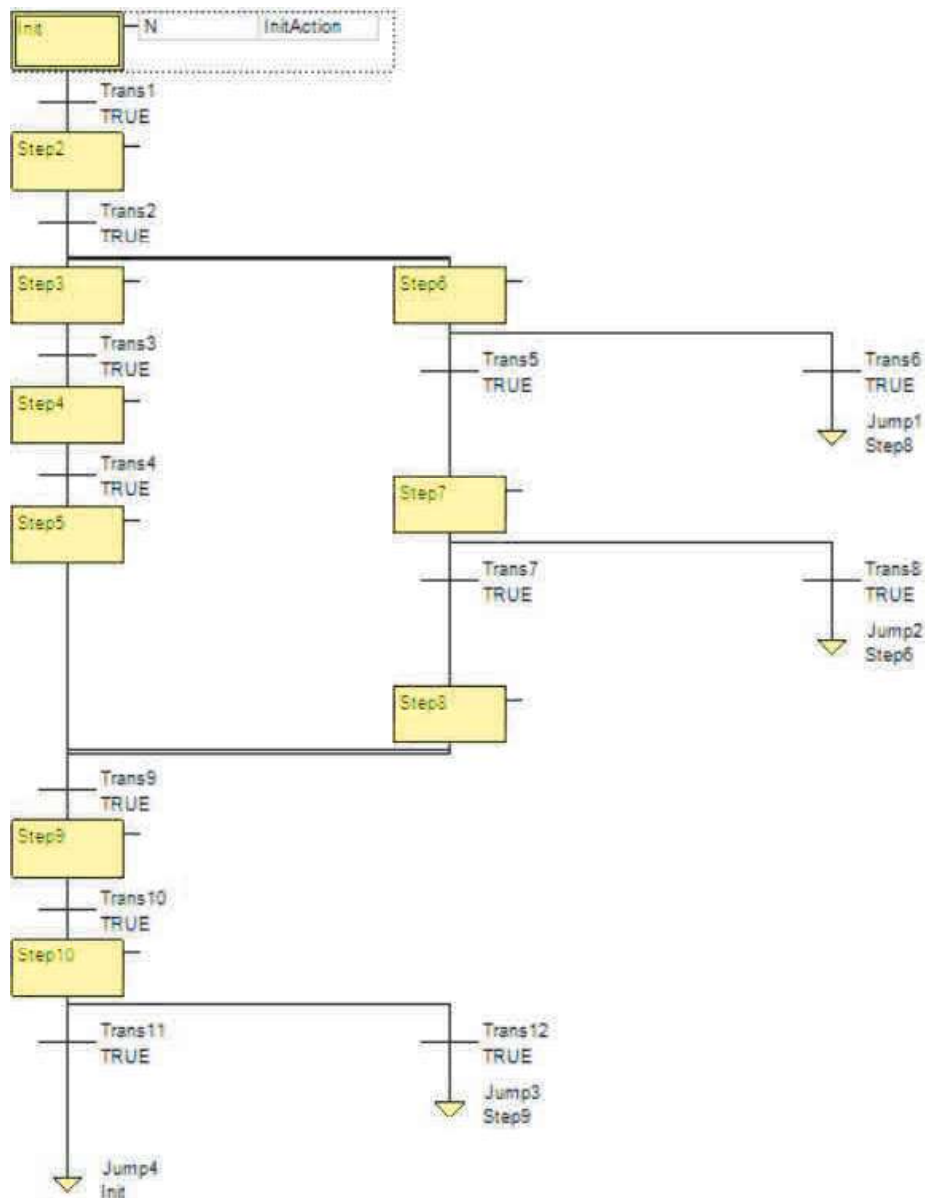


- To insert horizontal branch in the middle:
 - select *Trans11*
 - click *sequence selection* icon  to get branch with *Trans12* on the left
 - select *Trans12* and click *jump* icon  to get *Jump4* with *Step?*
 - double-click *Jump4* and change *Step?* to *Step9* in *Jump* properties window.



Renumbering

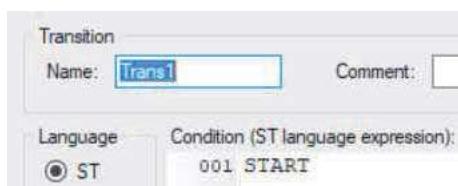
- Click the icon . Initial SFC diagram is thus obtained. Verify the diagram.



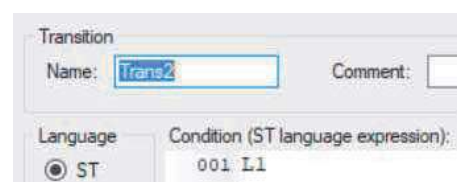
Completing the project

Conditions and actions

- Transition conditions
The conditions are shown in the target diagram. To implement the first one, double-click *Trans1* and write START in *Transition* properties window. Similarly for the other transitions.



Trans3: L4



Trans8: NOT L1

Trans4: Step4.T>T#10s

Trans5: L2

Trans6: Step5.X

Trans7: Step5.X

Trans9: TRUE

Trans10: Step9.T>T#3s

Trans11: NOT L3

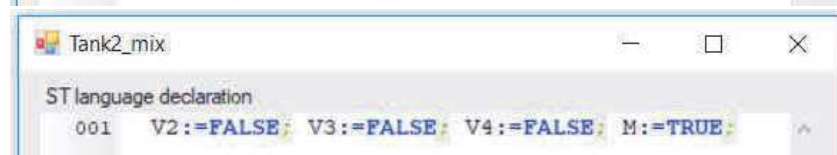
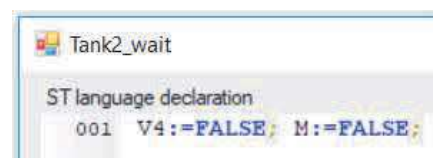
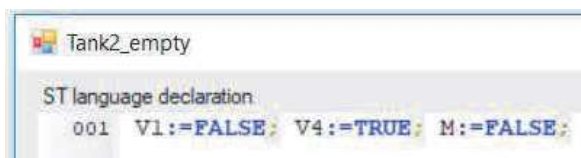
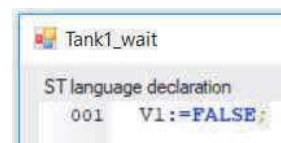
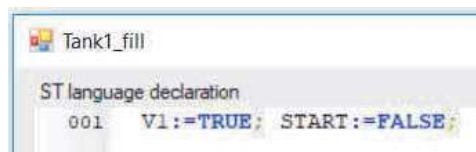
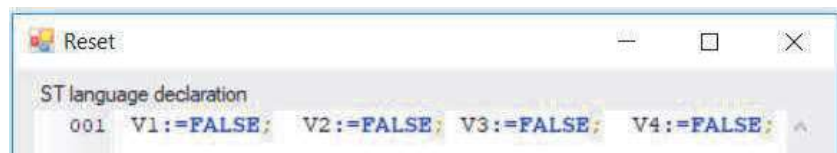
Trans12: L3 AND Step10.T>T#2s

- Actions

Change name *InitAction* to *Reset* in the action list, enter names of the other actions as in the target diagram.

Name	Lang.	Edit	Comment
Reset	ST	edit	
Tank1_fill	ST	edit	
Tank1_wait	ST	edit	
Tank2_empty	ST	edit	
Tank2_fill	ST	edit	
Tank2_mix	ST	edit	
Tank2_wait	ST	edit	

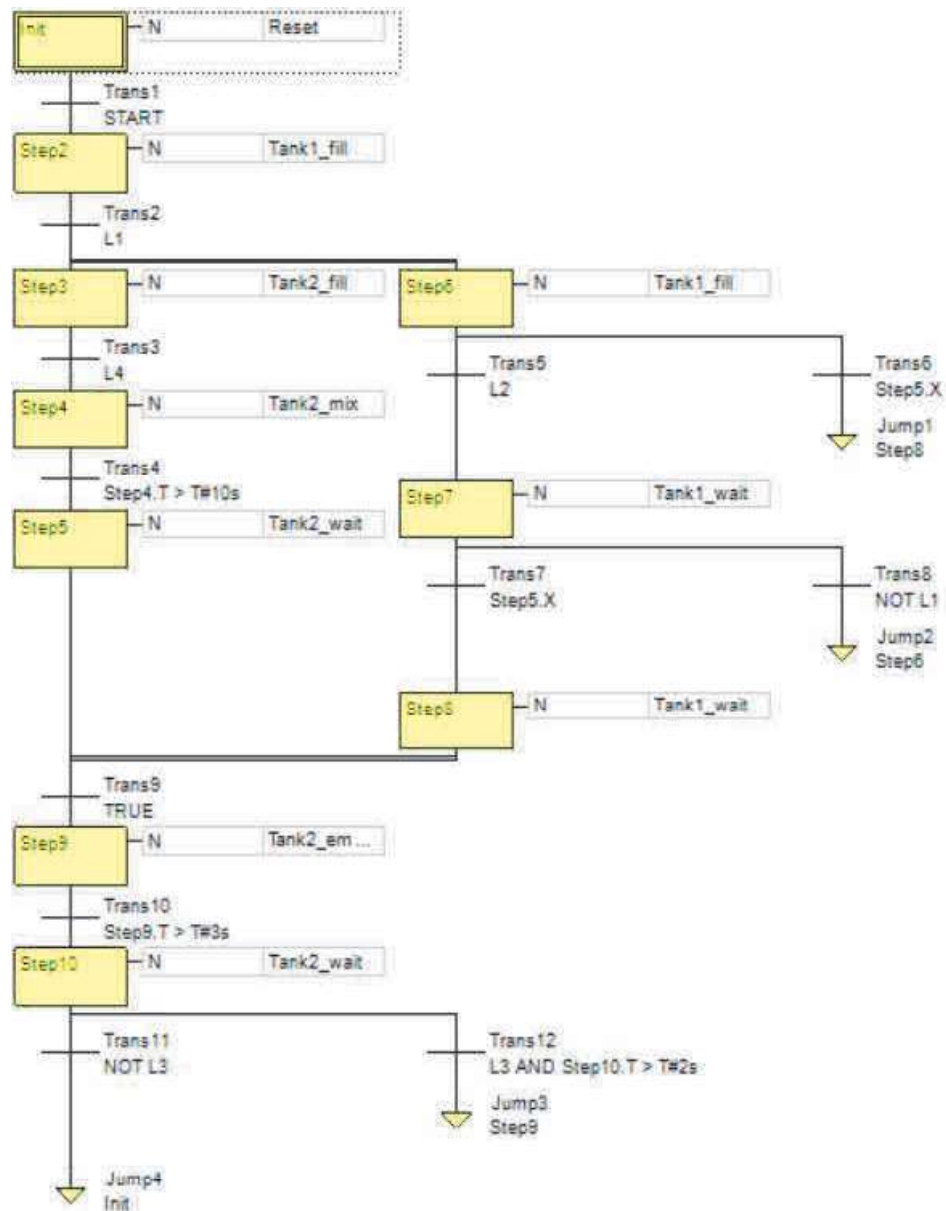
- Output assignments shown below are written in *Action* properties windows opened by pressing *edit* in the action list.



- Binding actions to steps

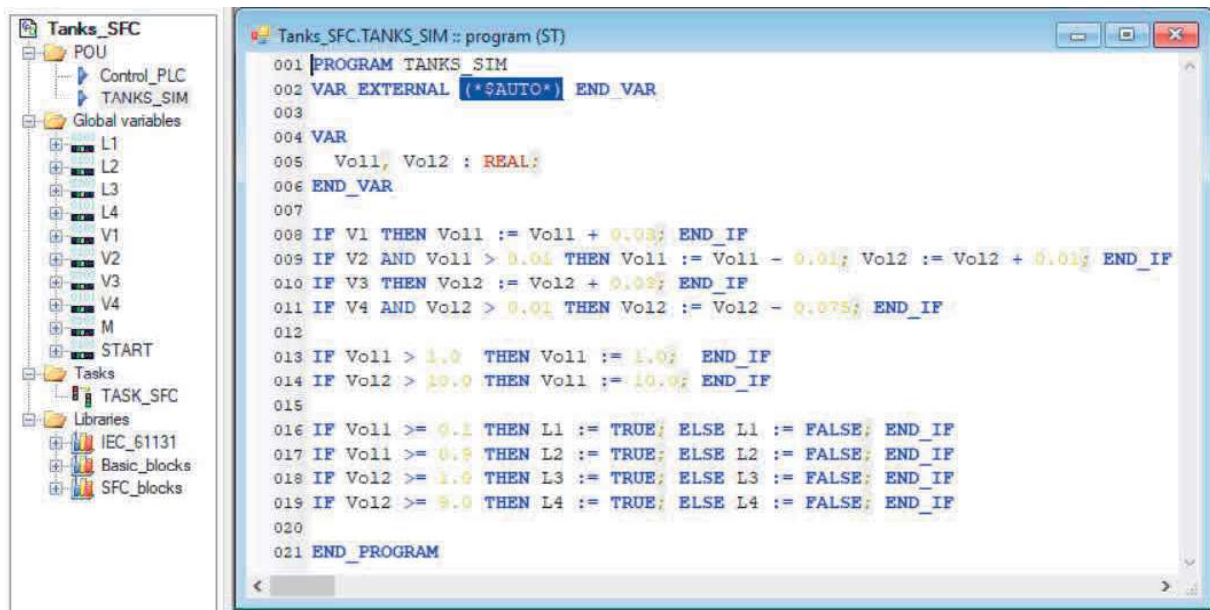
Double-click corresponding step in the diagram to open *Step/Action* properties window, press *Rename* (for *Init*) or *Add* (other steps) and choose appropriate action in appearing *Rename* window.

- This completes development of the SFC diagram. It is shown below for convenience.



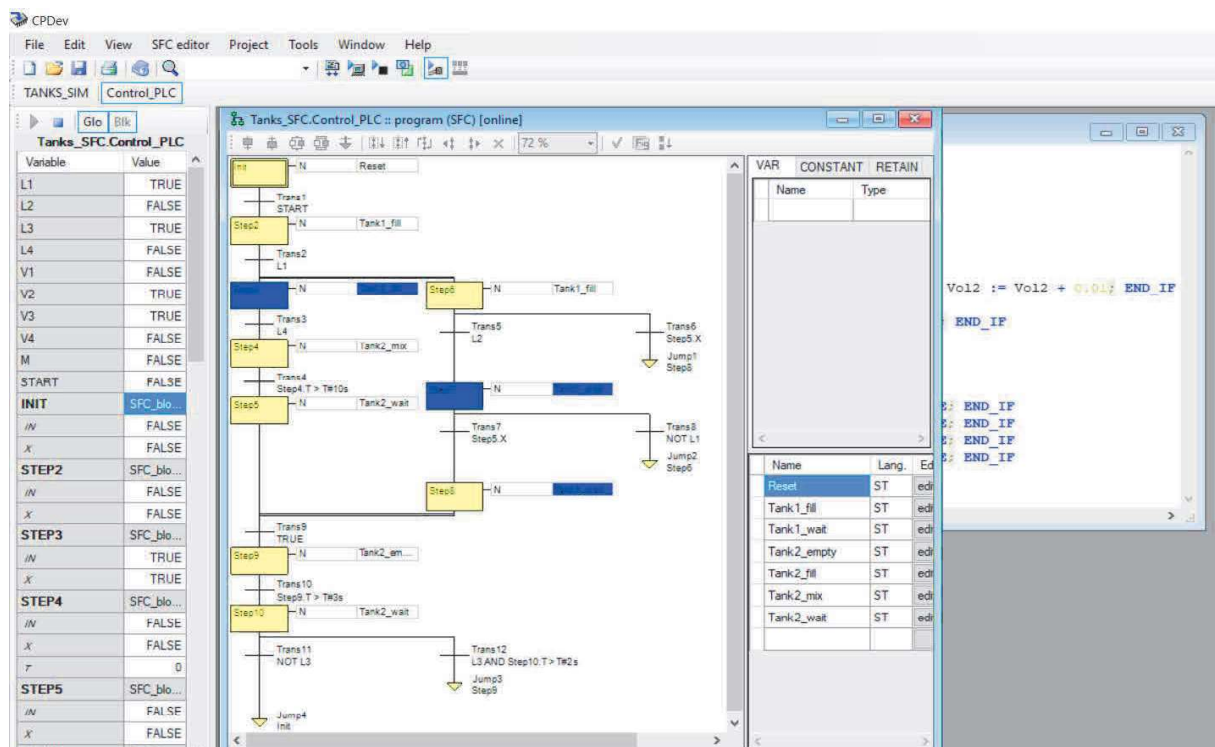
Volumes in tanks

- Tree of Tank_SFC project involves also Tank_sim program for calculation of volumes of liquids in the two tanks. The program is written in ST language and simulates filling and emptying in the simplest way, for visualization only.
- It is assumed that maximum volumes of the tanks are 1.0 and 10.0, respectively. Level sensors L1, L2 are at 0.1, 0.9, and L3, L4 at 1.0, 9.0. Opening the valves V1 to V4 increases or decreases the volumes by constant quantities (0.03, 0.01, 0.075).



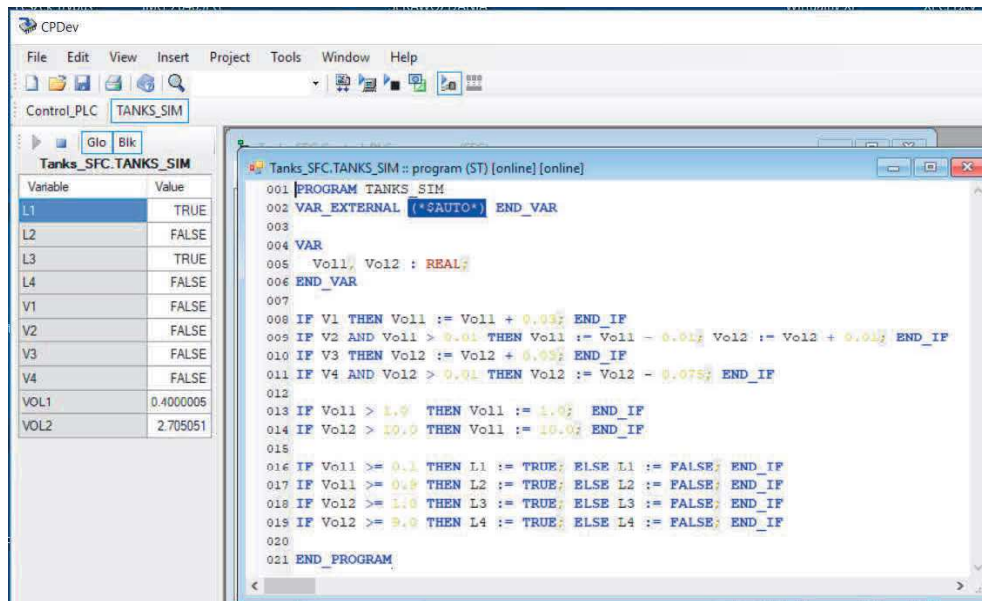
Simulation

- The project consists of two programs. In the online mode the variable list shown below corresponds to the SFC diagram currently on the top.

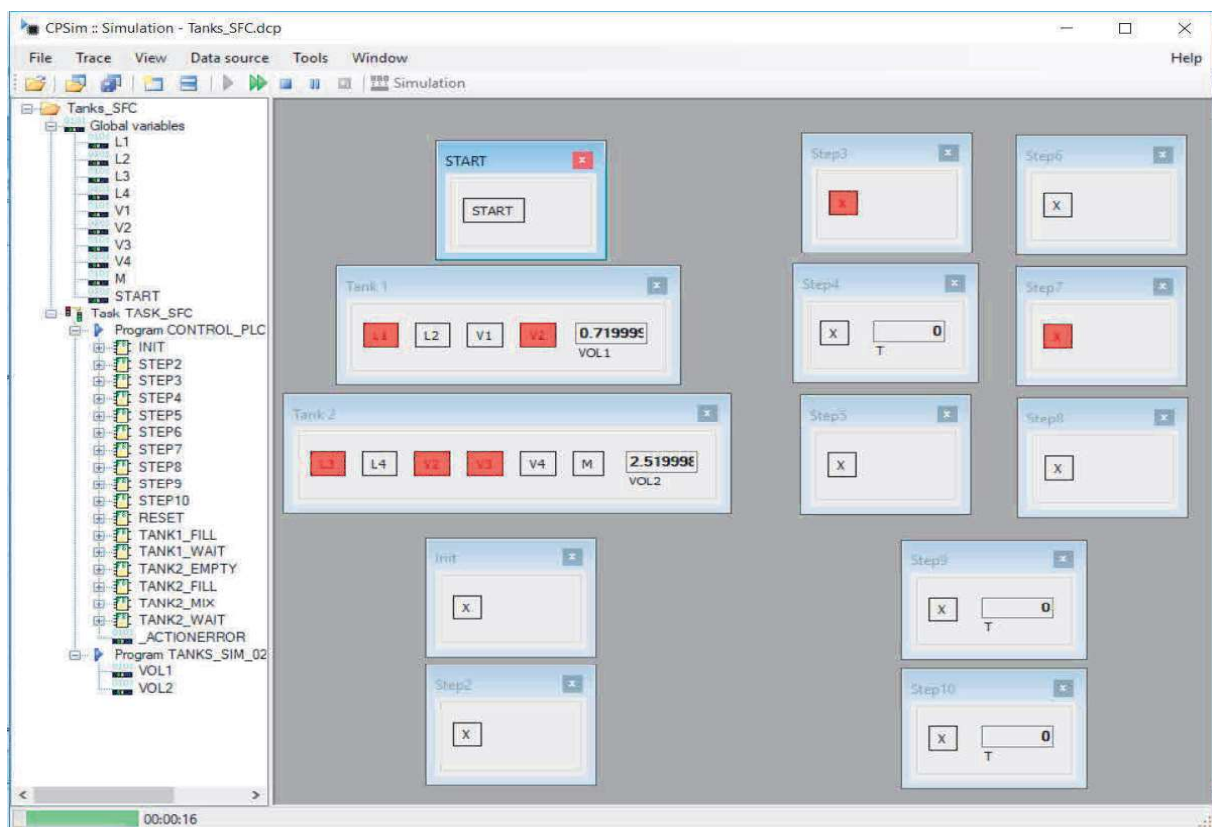


Remark. Online mode shows active steps and actions, independently of each other. Therefore the same *Tank1_wait* action is shown active both in Step7 and Step8, despite that Step8 is inactive.

- Placing the ST program on the top changes the variable list. Note volumes VOL1, VOL2 in the tanks (CPDev compiler changes lower-case characters into upper-case).



- In addition to the global variables, panels Tank1, Tank2 in CPSim window show volumes VOL1, VOL2 in the tanks. Step panels are arranged in three sections as in the SFC diagram.



Remark. Note that simultaneous sequence is active in this window, with Tank2 being filled and Tank1 waiting. Since volumes in the tanks change every cycle by constant quantities, time from START to finish depends on task cycle. For 200 ms cycle the time is about 45 seconds.

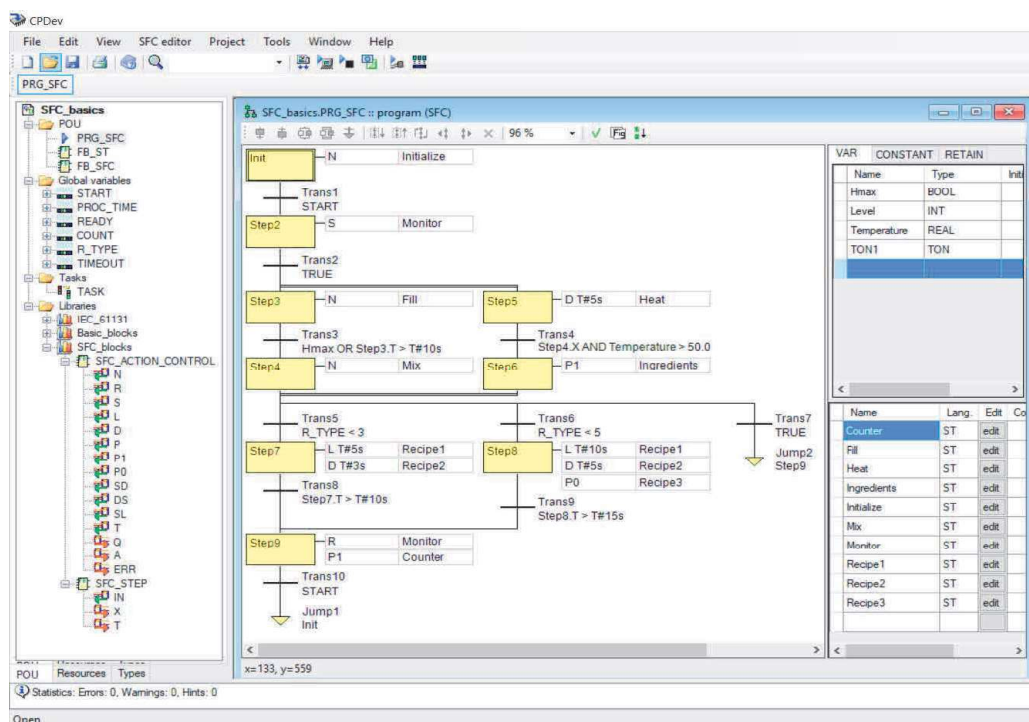
SFC editor summary

Elements and evolution rules

- Program or function block in SFC language may contain *steps*, *transitions*, *jumps*, *sequence selections* and *simultaneous sequences*.
- Step represents *actions* executed while the step is active.
- First step called *Init* is active upon initialization.
- State of a step can be monitored by two system variables, *step_name.X* and *step_name.T*. The first one is TRUE while the step is active, the second indicates time elapsed since activation of the step.
- Transition is determined by *condition* enabling passing to next step.
- Steps are separated by transitions, transitions are separated by steps.
- *Jumps* change execution order of step/transition sequences.
- *Sequence selection* chooses certain variant from a few sequences. It may be treated as a complex transition. Therefore it must be followed by a step.
- *Simultaneous sequence* runs a few concurrent sequences, and may be treated as a complex step. Therefore it must be followed by a transition.
- Passing from simultaneous sequence to next step requires evaluation of transition condition following the sequence to TRUE value. Besides, all steps preceding the transition must be *active*.
- Jumps outside a branch of simultaneous sequence should not be used because may lead to “unsafe” or “unreachable” SFC diagrams.

SFC_basics

- SFC_basics project shown below demonstrates additional capabilities of the editor. Physical meaning is not explained however. Focus is given on interpretation of qualifiers, local variables with instances of function blocks, and action blocks.



- The project consists of:
 - three POU: PRG_SFC program, FB_ST and FB_SFC function blocks
 - global variables: START, PROC_TIME, ..., TIMEOUT
 - task TASK_SFC
 - local variables: Hmax, Level, Temperature, TON1
 - actions: *Counter*, *Fill*, ..., *Recipe3*.

Inputs and outputs of SFC_ACTION_CONTROL and SFC_STEP blocks are listed in the project tree.

- Qualifiers
 - R, S* – *Monitor* action executed in *Step2*, then stored (*S*) with execution continued in other steps, and finally reset (*R*) in *Step9*
 - L* – *Recipe1* executed by limited (*L*) time (*T*) in *Step7* and *Step8*
 - D* – *Heat* executed with delay (*D*) time (*T*) in *Step5*; similarly *Recipe2* in *Step7* and *Step8*
 - P0* – *Recipe3* executed once (Pulse) at the end (falling edge) of *Step8*
 - P1* – *Ingredients* executed once (Pulse) at the beginning (raising edge) of *Step6*; similarly *Counter* in *Step9*.

See *Action control blocks* (below) or IEC 61131-3 standard for details on action qualifiers.

Local variables

- Upper-right part of SFC Editor window lists local variables (if any) used in actions and transition conditions.
- Tabs depend on POU type:
 - Program
 - VAR – local variable
 - VAR_CONSTANT – constant value
 - VAR_RETAIN – retentive variable
 - Function block
 - VAR – local variable
 - VAR_INPUT – input variable
 - VAR_OUTPUT – output variable
 - VAR_IN_OUT – input/output variable
 - VAR_CONSTANT – constant value
 - VAR_RETAIN – retentive variable
- New variable is entered by writing *Name* and selecting *Type* from appearing list. The list involves basic types, function blocks from *Libraries* and blocks defined in the project (FB_ST, FB_SFC here).
- To remove a variable, click the row header (to the left from *Name*) and press *Delete* key.

VAR			CONSTANT	RETAIN
Name		Type	Initial	
Hmax		BOOL		
Level		INT		
Temperature		REAL		
TON1		TON		

VAR_IN_OUT			CONSTANT	RETAIN
Name		Type	Initial	
X1		BOOL		
Y1		BOOL		

Remark. Function blocks defined in the project appear on the *Type* list provided that they are first saved after creating, followed by closing and opening the project in CPDev again.

Actions

- Lower-right part of *SFC Editor* window lists actions that may be bound to steps and executed during runtime.
- Actions are defined in textual ST language as sets of instructions with global and/or local variables.
- Each action can be invoked many times during single task cycle.
- New action is entered by writing *Name*, pressing *edit* and programming it in properties window.
- To edit previously created action, click *edit* or double-click any place in appropriate row.
- To remove an action, click the row header and press *Delete* key.

Name	Lang.	Edit	Col
Counter	ST	edit	
Fill	ST	edit	
Heat	ST	edit	
Ingredients	ST	edit	
Initialize	ST	edit	
Mix	ST	edit	
Monitor	ST	edit	
Recipe1	ST	edit	
Recipe2	ST	edit	
Recipe3	ST	edit	

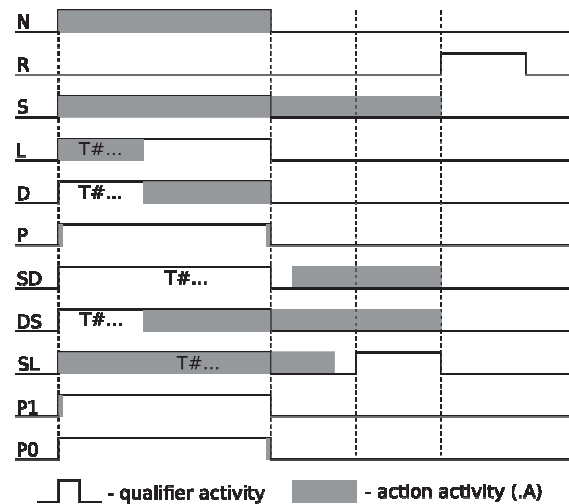
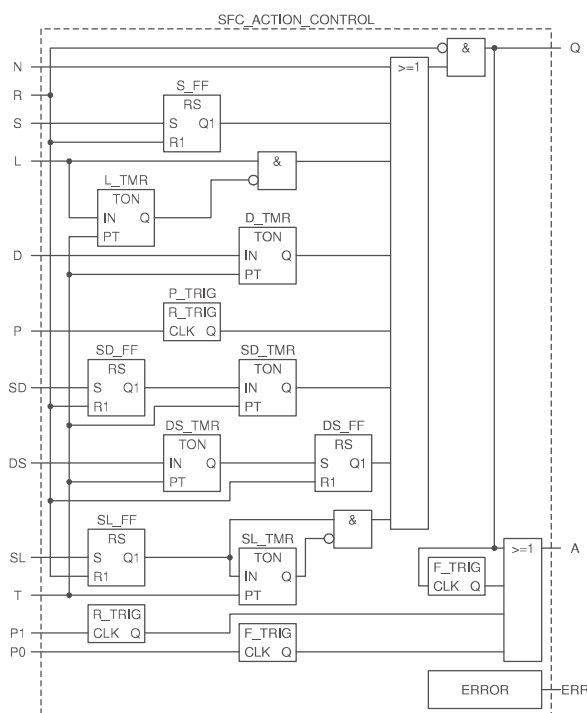
Action control blocks

- Action activities are controlled by instances of SFC_ACTION_CONTROL function block. Names of the instances are the same as names of the actions.
- Qualifiers *N*, *R* up to *SL* and parameter *T* of time qualifiers are inputs to action blocks (see below).
- Action control blocks have three outputs:
 - Q* — execution of the action indicated (TRUE)
 - A* — as *Q* but prolonged by one task cycle (see IEC 61131-3 for details)
 - ERR* — inconsistency of qualifiers assigned to the same action in different steps.
- Outputs of the action blocks can be used in programs of the actions and in transition conditions.
- FBD structure of SFC_ACTION_CONTROL block and time plots of the action qualifiers.

Monitor

```


ST language declaration
001 TON1 (IN:=Monitor.Q, PT:=T#120s);
002 TIMEOUT := TON1.Q;
003 PROC_TIME := TON1.ET;
  
```

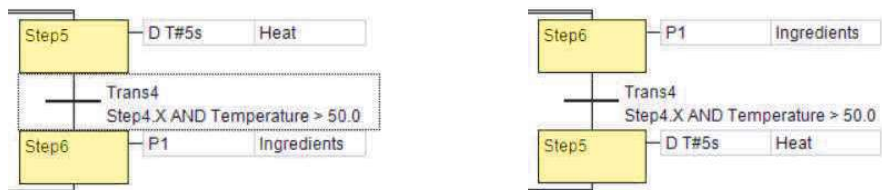




Editing SFC diagram

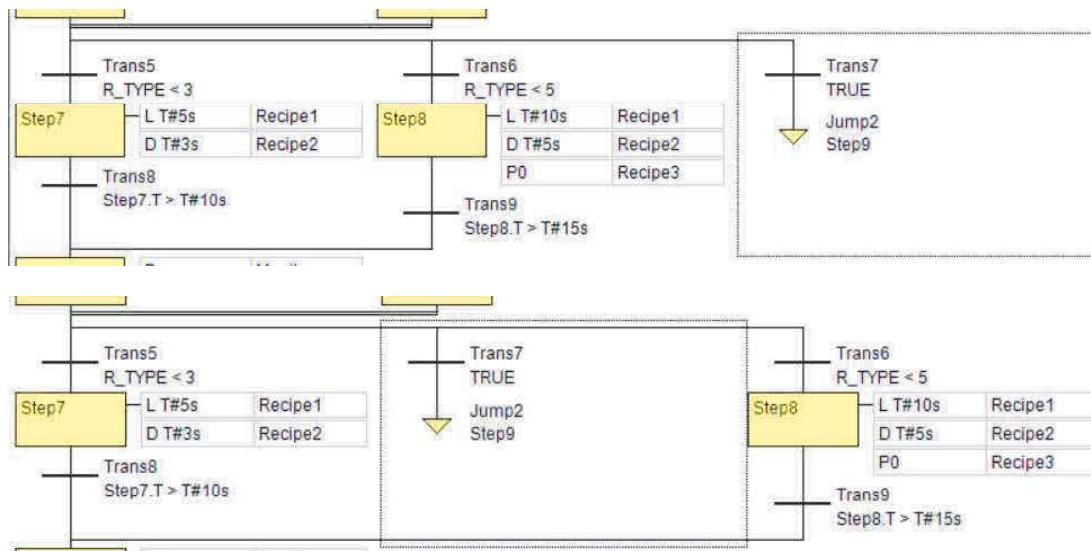
- New diagram for a program or function block consists of:
 - initial step *Init* with one empty *InitAction*
 - transition *Trans1* with condition TRUE
 - jump *Jump1* to *Init*.
 Variable list is empty. Action list includes *InitAction* only.
- To insert other elements into the diagram select appropriate step or transition and choose one of enabled icons in the toolbar or option of SFC editor.
- *Step + Transition* may be inserted:
 - before selected step or simultaneous sequence
 - after selected transition or sequence selection.
- *Transition + Step* may be inserted:
 - before selected transition or sequence selection
 - after selected step or simultaneous sequence.
- Branch of *Sequence selection* may be inserted on the right side of:
 - single transition
 - group of elements which begins and ends with a transition
 - another branch of sequence selection.
- Branch of *Simultaneous sequence* may be inserted on the right side of:
 - single step
 - group of elements which begins and ends with a step
 - another branch of simultaneous sequence.
- *Jump* may be inserted:
 - after the last transition of sequence selection branch.

Exchange and move



- *Exchange elements vertically*
 Three neighboring elements are involved, with upper and lower being exchanged around the middle one. The two elements must be of the same type, so *step*, *transition*, *sequence selection* or *simultaneous sequence*. Two groups with the same first and last elements can also be exchanged. To exchange:
 - select the middle element
 - click *exchange vertically* icon  or choose such option of the editor.
 Exchange of steps separated by transition is shown below (from SFC_basics project).



- *Move to the left or right*
 Branch of *sequence selection* or *simultaneous sequence* can be moved to the left or right. To move:
 - select required branch by clicking horizontal line above or below the branch
 - click one of the icons ,  or choose editor option.

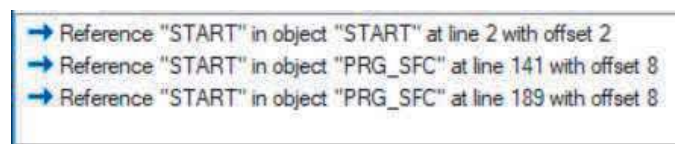


Remark. Sequence selection branches are evaluated from left to right, so moving elements affects execution.

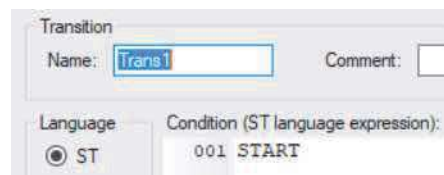
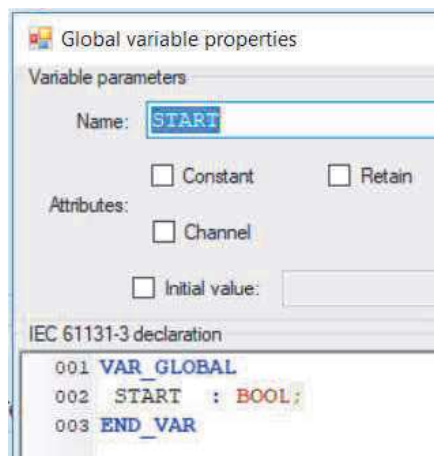
- *Move up or down*
Pairs *step + transition* and similar ones indicated in *Delete pairs* (see *Diagram corrections*) can be moved up or down. A few such pairs can be moved as well. Moving proceeds similarly as above by means of the icons , .

Global variable search

- Enter name of global variable in *Global Search* cell (top-right corner of CPDev window).
List of references to the variable appears at the bottom message window.

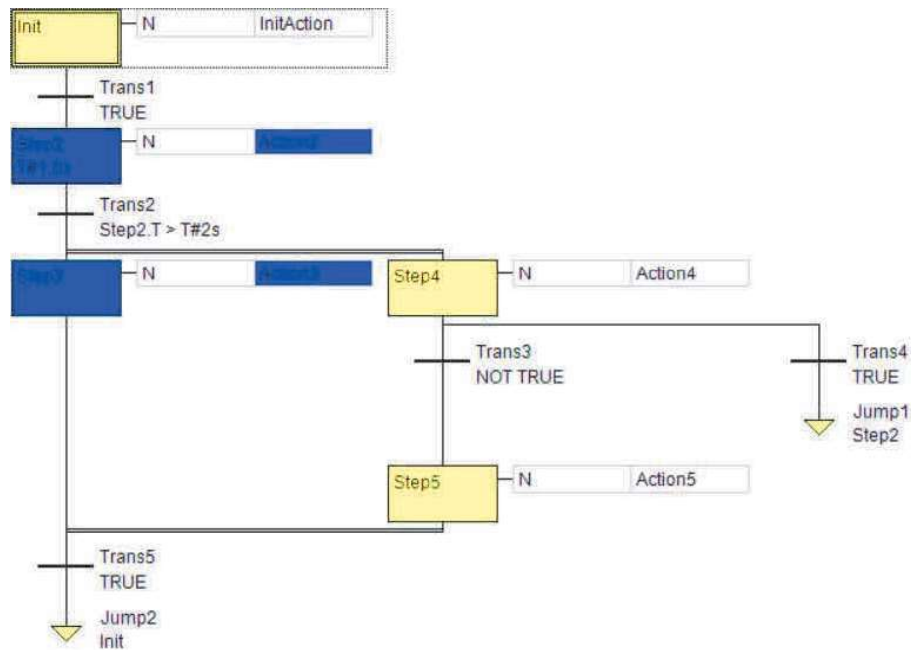


- Double-click the row in the list opens the reference window.



Unreachable and unsafe SFC

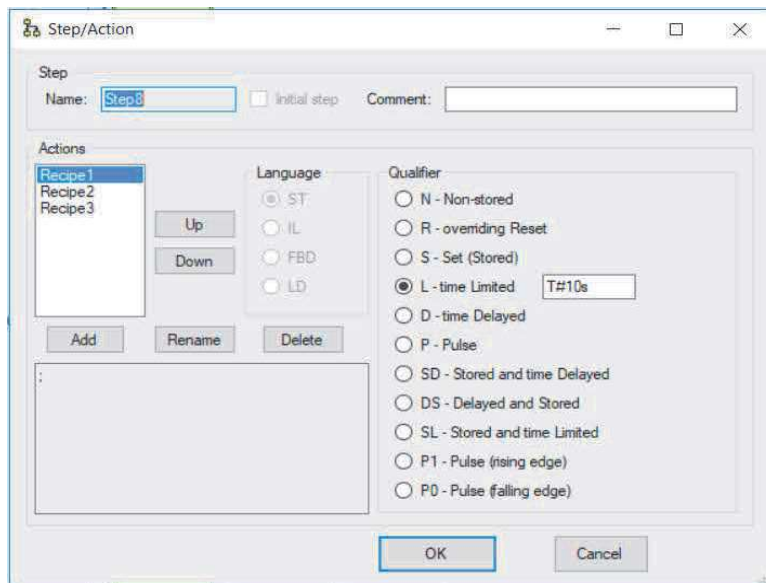
- Diagram in which at least one step can never be activated is called *unreachable*. It typically occurs in case of incomplete branches or jumps outside of simultaneous sequences. SFC editor partially prevents this by allowing to add complete branches only, but jumps are not forbidden.
- Diagram in which a step can be activated while it is still active is called *unsafe*, since it may lead to unpredictable behavior of the control system. Unsafe diagrams may be created by the editor, so the user should take care to avoid them. The diagram shown below in online mode is unsafe, since *Jump1* activates *Step2* while leaving *Step3* active (not deactivated, still running after enter into simultaneous sequence).



Properties windows

Step/Actions

- *Step/Action* properties window allows to bind action to step, change action qualifier and add a comment. Double-click relevant step in the diagram to open the window.

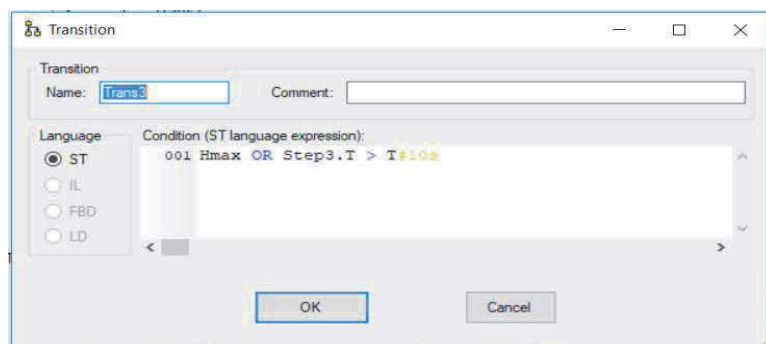


- Click *Add* to add new action to *Actions* binding list in the window. Small window opens (*Add action*) in which action name may be selected from action list previously defined in the main window (lower-right corner).
- *Language* of selected action cannot be changed (read-only). It is set automatically as ST while editing the action.
- *Qualifier* states how long the action is active during runtime. For instance: *N* – all time while the step is active, *S* – until another step with *R* qualifier invokes the same action, *SL* – in limited time no matter how long the step is active. See IEC 61131-3 standard for details.
- Buttons:
 - *Up/Down* change invocation order of actions.
 - *Rename* renames selected action (like *Add*, but without adding to the list).
 - *Delete* removes selected action from the binding list.

Remark. If the same action with time-dependent qualifier, such as L, D, SD, DS or SL, is activated at the same moment at two different steps (or twice at the same step), it appears in red while in the online mode (qualifier conflict). The ERR output of the action becomes TRUE.

Transition

- *Transition* properties window allows to rename the transition, add a comment and edit transition condition. Double-click the transition to open the window.



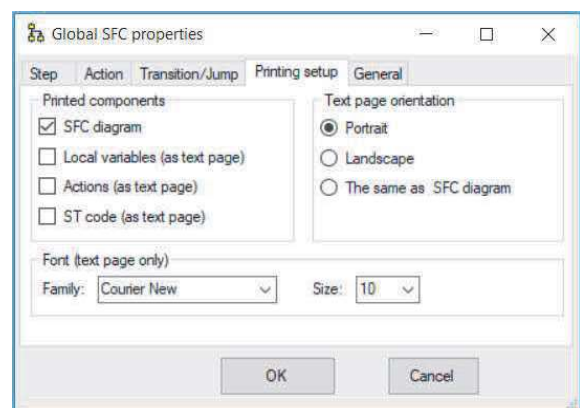
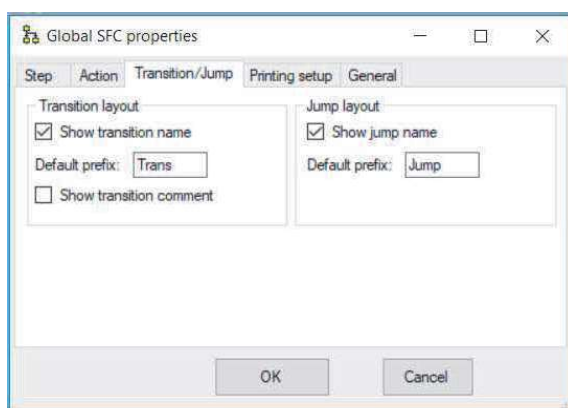
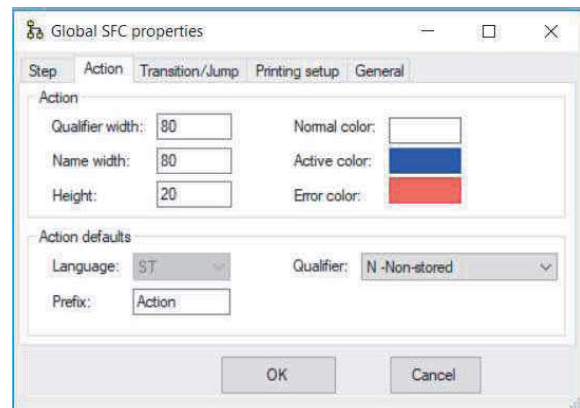
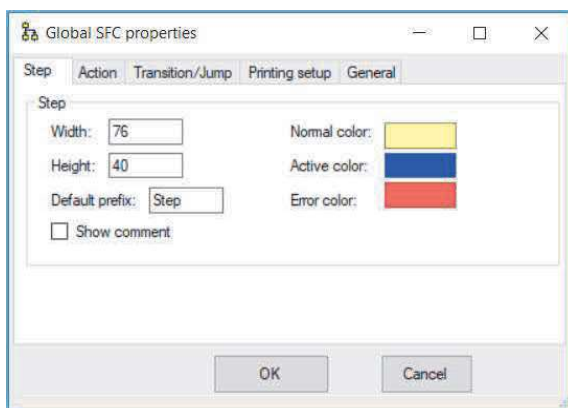
Jump

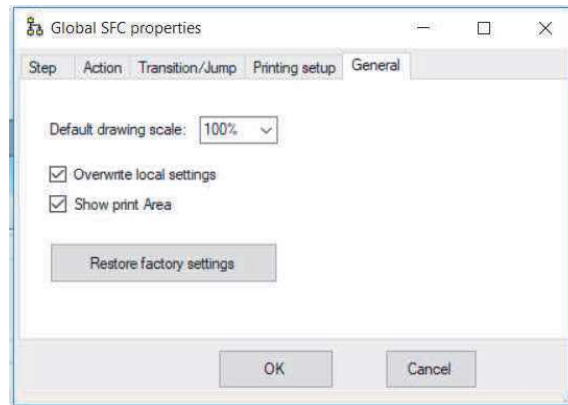
- *Jump* properties window allows to rename the jump and indicate name of next step. Double-click the jump to open the window.



Global SFC properties

- Selecting *Tools > Modules > SFC configuration* opens *Global SFC properties* window with five tabs, *Step*, *Action*, *Transition/Jump*, *Printing setup* and *General*. The properties are *global*, so they apply to all SFC diagrams of the project.
- *Step* and *Action* tabs specify layout settings of the element, such as size, colors, prefix, etc. *Transition/Jump* defines prefixes and selects what to show in the diagram.





- *Printing setup* specifies printed components, so the diagram, local variables, actions and ST code. Orientation of text pages can be also selected (diagram orientation is selected in *Printing*).
- *General* tab determines drawing scale by displaying print line on the diagram and restores factory settings. After changing the settings reload of SFC window is required. Local settings are the ones previously saved in a file with the diagram.

Other issues

Changing SFC program name

Program name given initially needs to be changed sometimes in the final version.

- To change program name:
 - open the diagram (if closed)
 - select the program in the project tree
 - choose *Project* → *Items* → *Rename*
 - enter new name in the tree
 - accept (*Enter*)
 - build and save the project.

Library warnings

When library is linked to a project, CPDev automatically stores library version, timestamp and number of compiler version which created the library (LCP file). When the project is open again and built, those markers are compared with markers of actual library and compiler. Inconsistencies trigger warnings.

- The following warning indicates outdated library:

 Declared library timestamp date is not equal to timestamp found in library name from file "C:" location

If library content and compiler version have not been changed, ignore the warning and proceed with the project. Warning will not appear on next opening.

Otherwise:

- remove the library from the project tree by selecting the library and choosing *Remove* from context menu
- import actual library into the project by *Project* → *Import* → *Library* from appropriate file.
- New version of CPDev compiler includes recompiled system libraries, i.e. *IEC_61131.lcp* and *Basic_blocks.lcp*. So only user library, if any, needs recompilation, export as LCP file into given location (*Project* → *Export* → *Library*) and import into the project as above.

Automatic save

- Every 10 minutes (default) CPDev creates a file in **temporary directory** (see operating system settings) called *CPDev_ProjectNameOrFileName_EightHexNumbers.xml* with source code of the project. This is a back-up in case of CPDev crash.
- If SFC diagram has been changed within that period, it is automatically verified (if open).
- Automatic save period may be changed in *Environment options* → *Editing*.

Printing

Size adjustment

- Grey lines partition total work area of the editor into printed pages. Final size of the diagram may be decided before printing to fill in the pages conveniently.

Page setup

- Choose *Tools* → *Environment options* → *Configuration* → *Page setup*.

Configuration

Projects Editing Colors Miscellaneous Compiler Help Page setup

Border margins

Left (cm): 0.5 Right (cm): 0.5

Top (cm): 0.5 Bottom (cm): 0.5

☒ Draw border Padding (cm): 0.3

Paper/Orientation

Paper size: A4

☒ Portrait ☐ Landscape

Printing scale

User scale

User scale (%): 100

☒ Draw title block

Title block fields

☒ Subject ☒ Company

☒ Project name ☒ Project version ☒ POU name ☒ Author

☒ Filename ☒ Date ☒ Page number

☒ Update settings upon printing session

OK Cancel

- Upper part of the options involves *Border margins*, *Paper size*, *Orientation* and *Printing scale* (independent of the editor scale).
- Number of pages being printed is determined automatically according to the size of the diagram and printing scale (user selected). Small diagrams need one page, large ones a few.
- Lower part specifies what fields appear in the title block (table) printed at the end of each page. Table changes somewhat if some fields are not selected.

Subject: Program keeps temperature in a greenhouse within a range		Company: Rzeszow University of Technology	
Project: Greenhouse_SFC	Version: v.1	POU: PRG_Greenhouse	Author: Jan Nowak
File: C:\Users\Leszek Trybus\Documents\CPDev - programy, instrukcje\Projekty...		Date: 26.04.2018 11:31	Page: 1 / 1

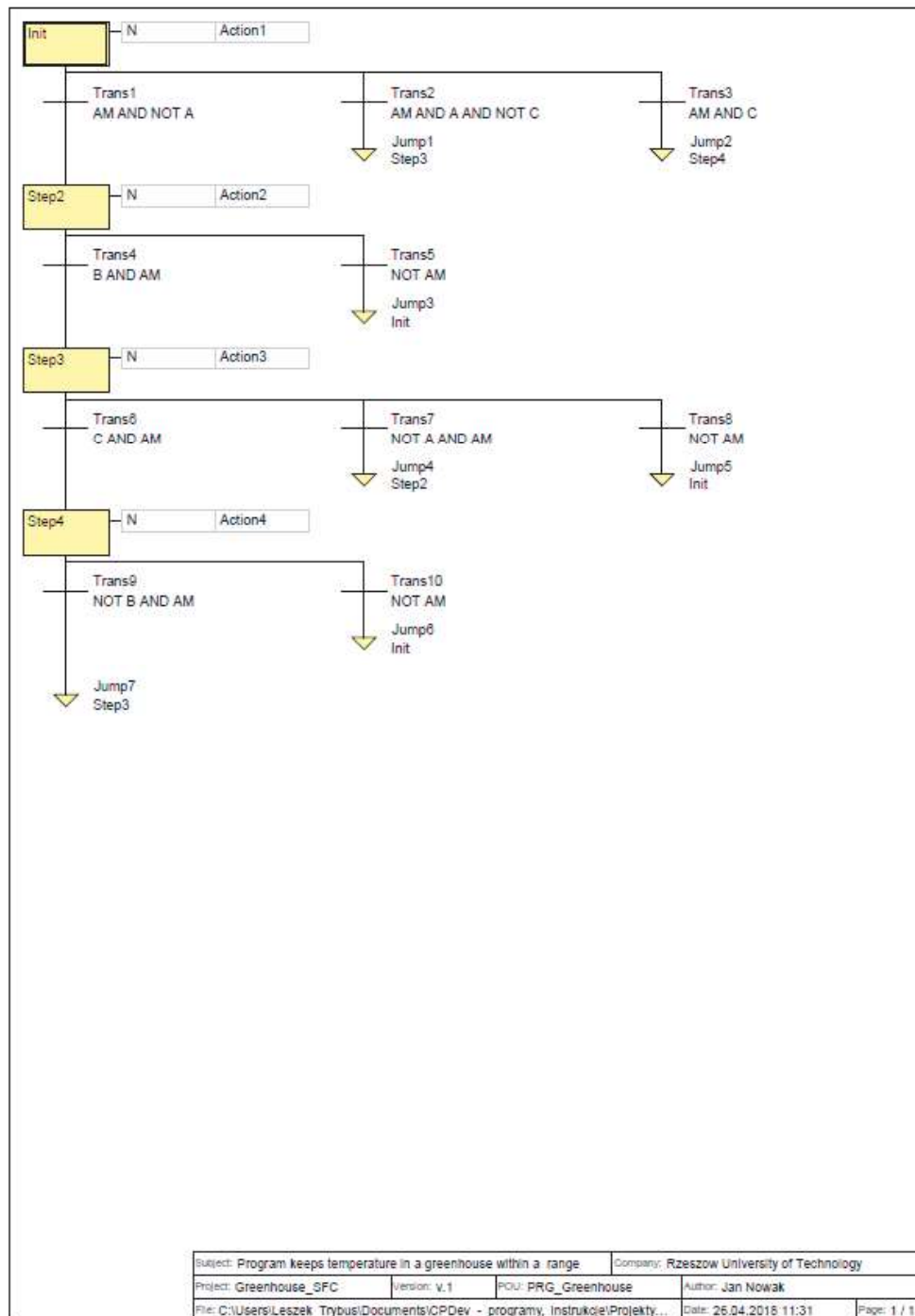
- Data for *Subject*, *Company*, *Project*, *Version*, *Author* and *File* are copied from *Project properties* window. Active window determines *POU* name.

Remark. Options of *Page setup* are kept in CPDev database (*Environment options*), so they apply to each printing until changed.

Print

- Choose *File* → *Print* in the main menu or press **Ctrl+P** keys or icon in the toolbar (direct print).
- Select printer and preferences in printer selection window. Change of preferences and printout adjusts grey lines and *Page setup* options accordingly.
- Press *Print*.

Remark. Printing to virtual printer such as Microsoft Print to PDF or PDF Creator is convenient way to verify expected printout. Vertical orientation (portrait) suits LD, SFC and horizontal (landscape) FBD.



Remark. If question *Do you want to print actions code?* is accepted, the ST code is printed.