Bending machine 24V

Plan and implement automatic processes

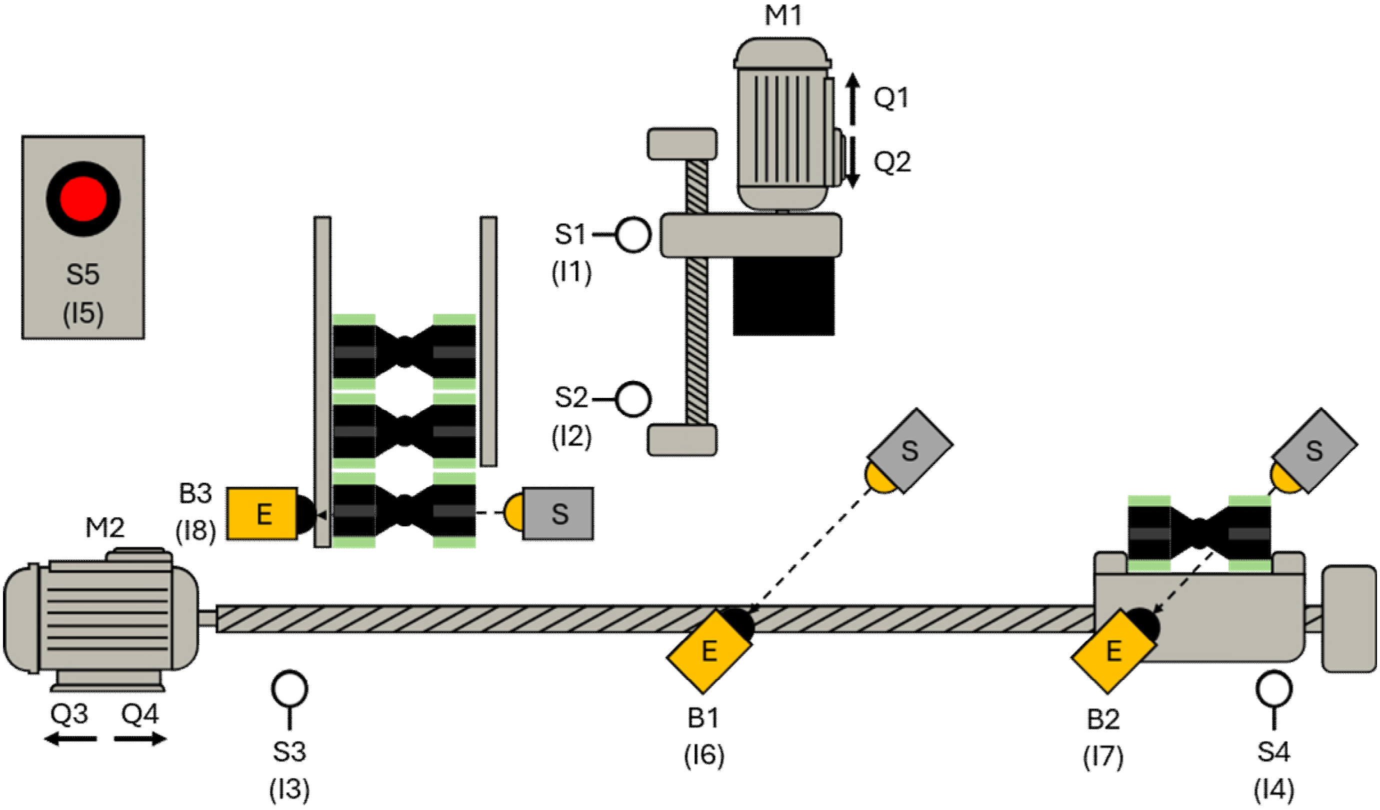


Table of contents

10 Plan and implement automatic processes

10.1 Exercise: Planning the process chain with GRAFCET

Target:

I can create a GRAFCET sequence chain for the bending machine using the functional description.

Task:

Create a sequence chain for the bending machine in GRAFCET so that the function is implemented using the function description from the "Model" chapter.

Function:

1. Initial step
   No actions are performed in the initial step.
   The step chain remains in this step until the start button (S5) is pressed (positive edge) and the magazine is not empty at this point (B3).

2. Move press to home position
   After starting the system, the bending machine should first be moved to the home position. For this purpose, the motor M1 is controlled with Q1. To prevent the motor from jamming, it may only be activated as long as the end position S1 has not been reached.
   If the press is in the upper end position (S1), you can switch to the next step.

3. Move the transport carriage to the home position
   Now that the press is in the home position, the transport section must also be moved to the home position. To do this, the motor M2 is controlled with Q4. It may only be activated if the end position S4 has not been reached.
   If the transport section is in the home position (S4) and there is no workpiece on the carriage (B2), you can switch to the next step.

4. Retrieve workpiece from magazine
   To pick up a workpiece from the magazine, the motor M2 must be controlled in direction Q3. Actuation may only take place if the end position S3 has not been reached.
   If the transport carriage is below the magazine (S3), you can switch to the next step.

5. Move workpiece into press
   The workpiece must now be moved into the press. Motor M2 must be controlled in direction Q4. This may only take place as long as the end position S4 has not been reached.
   If light barrier B1 is interrupted by the workpiece during the movement, the carriage must stop and move to the next step.

6. Bend workpiece
   The workpiece is bent by moving motor M1 in direction Q2. Downward actuation may only take place as long as the end position S2 has not been reached.
   If the press is in the lower end position (S2), you can switch to the next step.

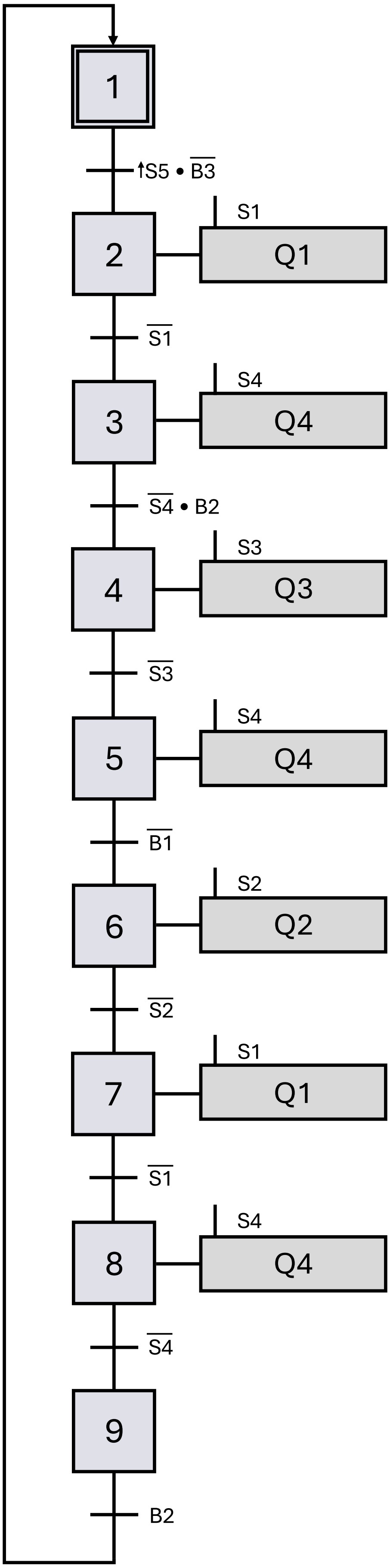
7. Reduce press
   After bending, return the press to the home position.
   Motor M1 is controlled in direction Q1. The motor may only be controlled as long as the end position S1 has not been reached.
   If the press is in the upper end position (S1), you can switch to the next step.

8. Move workpiece to removal position
   The bent workpiece must be moved forwards to the removal position at the end. Motor M2 is controlled in direction Q4. Actuation may only take place as long as the end position S4 has not been reached.
   If the transport section is in the front end position (S4), you can switch to the next step.

9. Waiting for acceptance
   No actions are performed in this step. Once the workpiece has been removed (B2), you can return to the initial step.

Soluti

Solution:


1
↑S5 • $\overline{B3}$
2
S1
Q1
$\overline{S1}$
3
S4
Q4
$\overline{S4}$ • B2
4
S3
Q3
$\overline{S3}$
5
S4
Q4
$\overline{B1}$
6
S2
Q2
$\overline{S2}$
7
S1
Q1
$\overline{S1}$
8
S4
Q4
$\overline{S4}$
9
B2

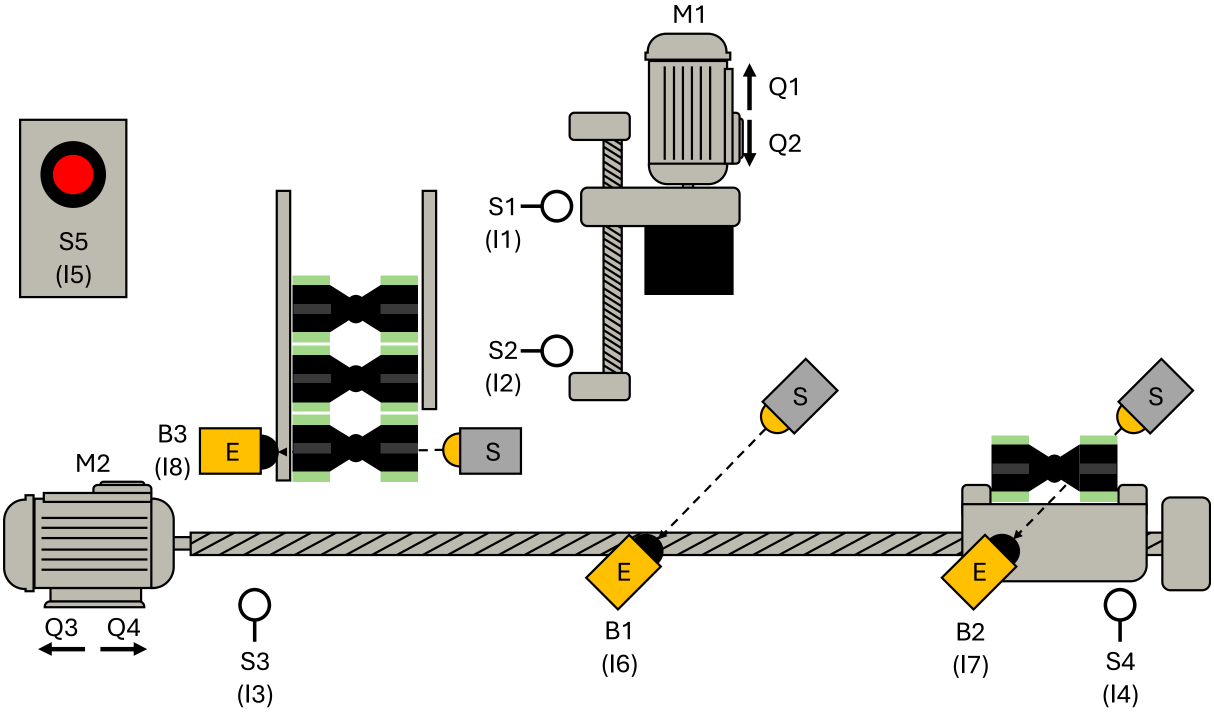
10.2 Exercise: Convert GRAFCET sequence chain into program code [FUP].

Target:

I can create an automatic program using the function description and the sequence chain created in GRAFCET.

Task:

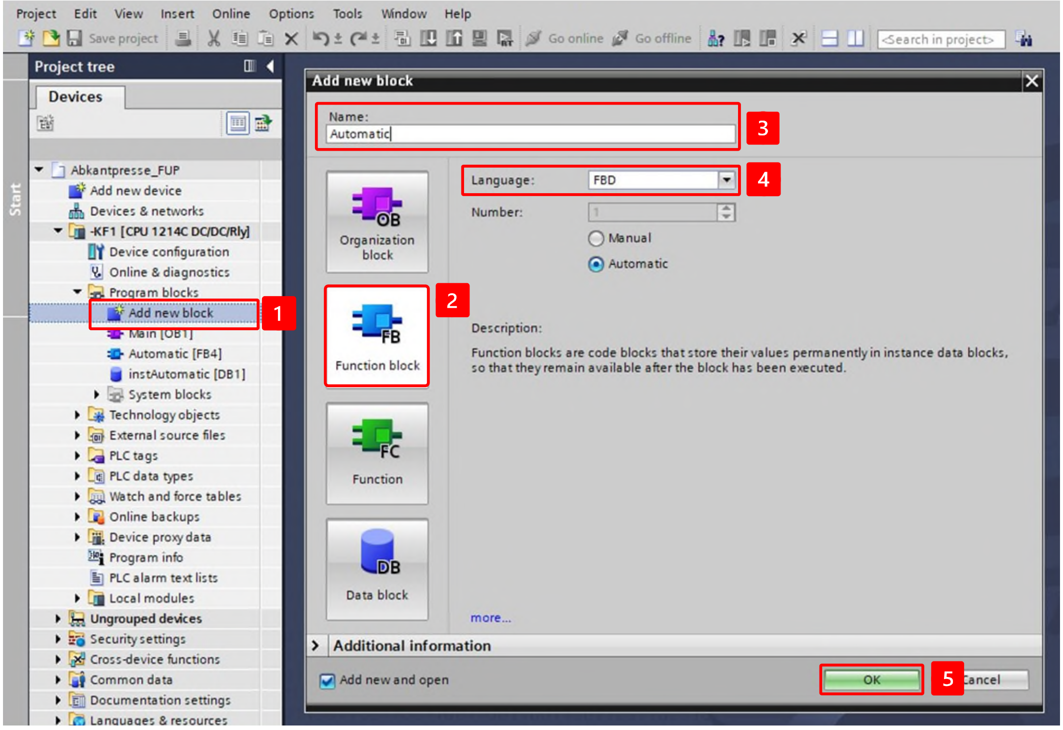
Create the PLC program for the automatic sequence for the bending machine so that the function is implemented based on the system description.



Picture 1 System diagram

Procedure:

1. Create a new function block, select the desired programming language and assign a meaningful name:



Name: Automatic

Language: FBD

Number: 1

○ Manual

● Automatic

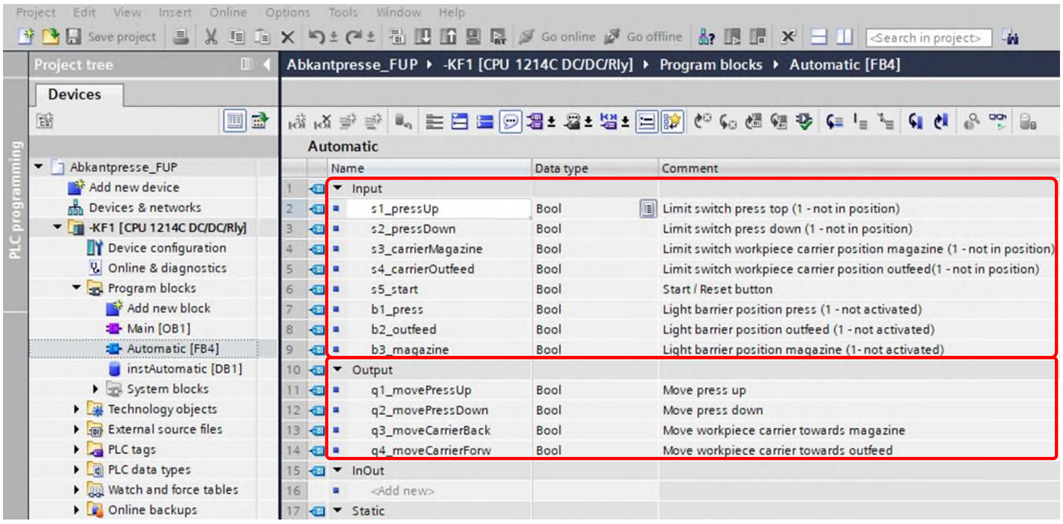
Description:

Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available after the block has been executed.

more...

☑ Add new and open

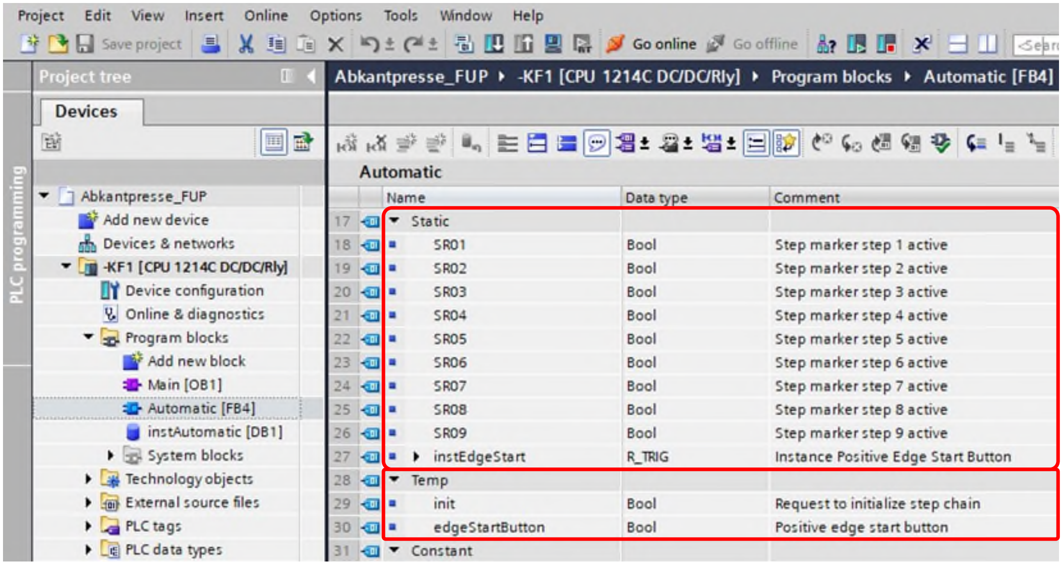
2. Declare variables for the sensors and actuators in the function block interface:


Abkantpresse_FUP ▸ -KF1 [CPU 1214C DC/DC/Rly] ▸ Program blocks ▸ Automatic [FB4]

Automatic

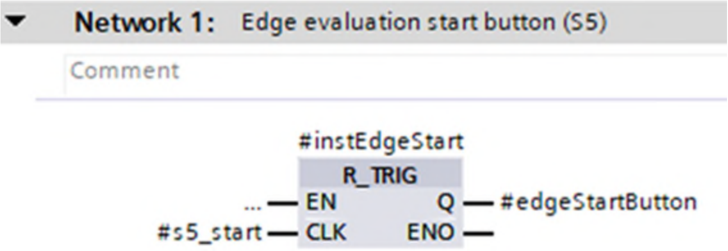
	Name	Data type	Comment
1	▼ Input		
2	s1_pressUp	Bool	Limit switch press top (1 - not in position)
3	s2_pressDown	Bool	Limit switch press down (1 - not in position)
4	s3_carrierMagazine	Bool	Limit switch workpiece carrier position magazine (1 - not in position)
5	s4_carrierOutfeed	Bool	Limit switch workpiece carrier position outfeed(1 - not in position)
6	s5_start	Bool	Start / Reset button
7	b1_press	Bool	Light barrier position press (1 - not activated)
8	b2_outfeed	Bool	Light barrier position outfeed (1 - not activated)
9	b3_magazine	Bool	Light barrier position magazine (1- not activated)
10	▼ Output		
11	q1_movePressUp	Bool	Move press up
12	q2_movePressDown	Bool	Move press down
13	q3_moveCarrierBack	Bool	Move workpiece carrier towards magazine
14	q4_moveCarrierForw	Bool	Move workpiece carrier towards outfeed
15	▼ InOut		
16	<Add new>		
17	▼ Static		

3. Declare the step flags in the static area of the function block interface, an instance for the edge evaluation of the start button, as well as a variable for initialization and for the positive edge of the start button in the temporary area:



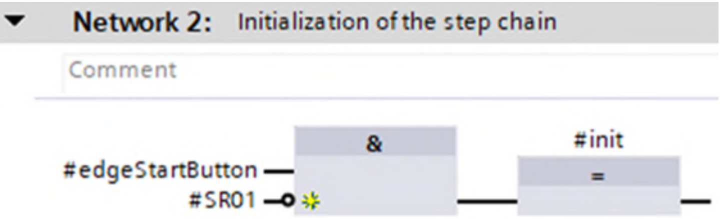
	Name	Data type	Comment
17	▼ Static		
18	SR01	Bool	Step marker step 1 active
19	SR02	Bool	Step marker step 2 active
20	SR03	Bool	Step marker step 3 active
21	SR04	Bool	Step marker step 4 active
22	SR05	Bool	Step marker step 5 active
23	SR06	Bool	Step marker step 6 active
24	SR07	Bool	Step marker step 7 active
25	SR08	Bool	Step marker step 8 active
26	SR09	Bool	Step marker step 9 active
27	▸ instEdgeStart	R_TRIG	Instance Positive Edge Start Button
28	▼ Temp		
29	init	Bool	Request to initialize step chain
30	edgeStartButton	Bool	Positive edge start button
31	▼ Constant		

4. Program the evaluation of the positive edge of the start button (S5) in the first network:


Network 1: Edge evaluation start button (S5)

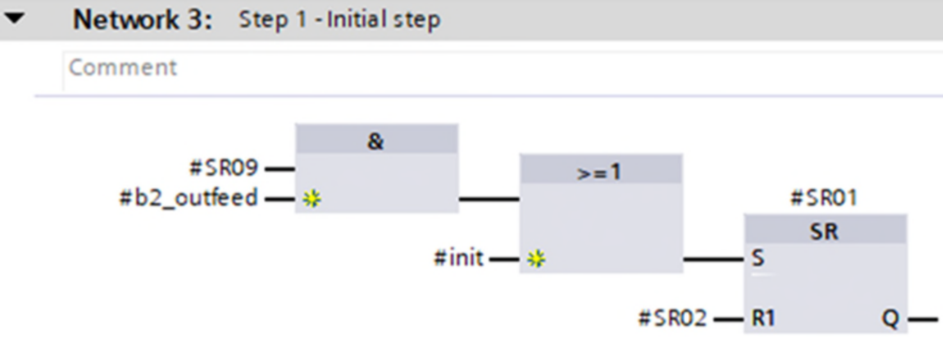
Comment

5. Program the initialization of the chain in the next network. The function description indicates that the step chain should be initialized when the start button is pressed (positive edge) and the chain is not in the initial step (SR01):


Network 2: Initialization of the step chain

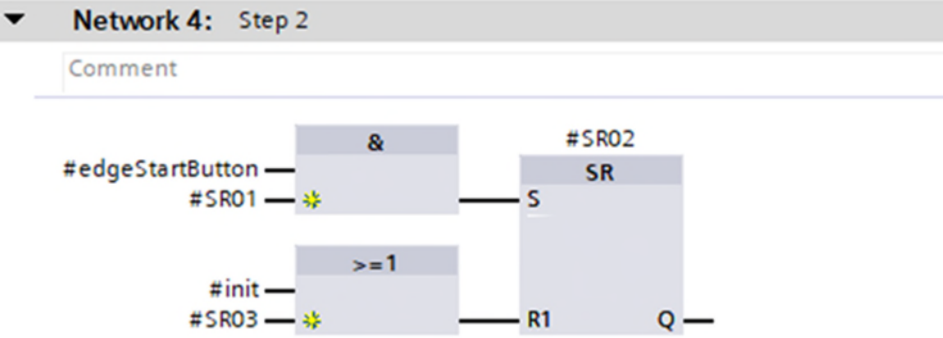
Comment

6. In the following networks, implement the individual steps using flip-flops based on the GRAFCET. A new network must be used for each step:



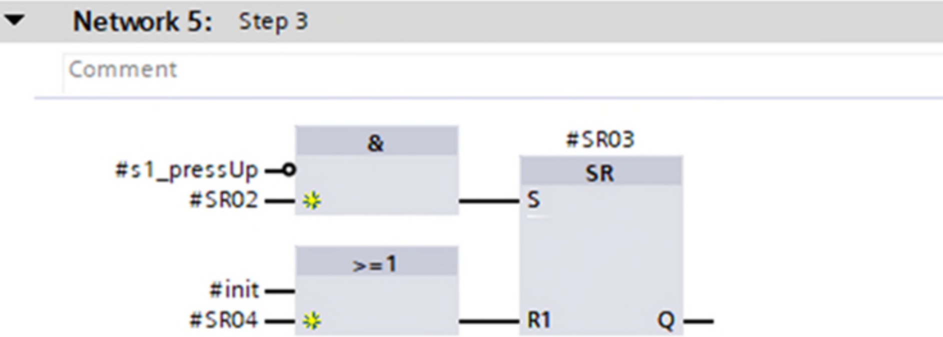
Network 3: Step 1 - Initial step

Comment



Network 4: Step 2

Comment

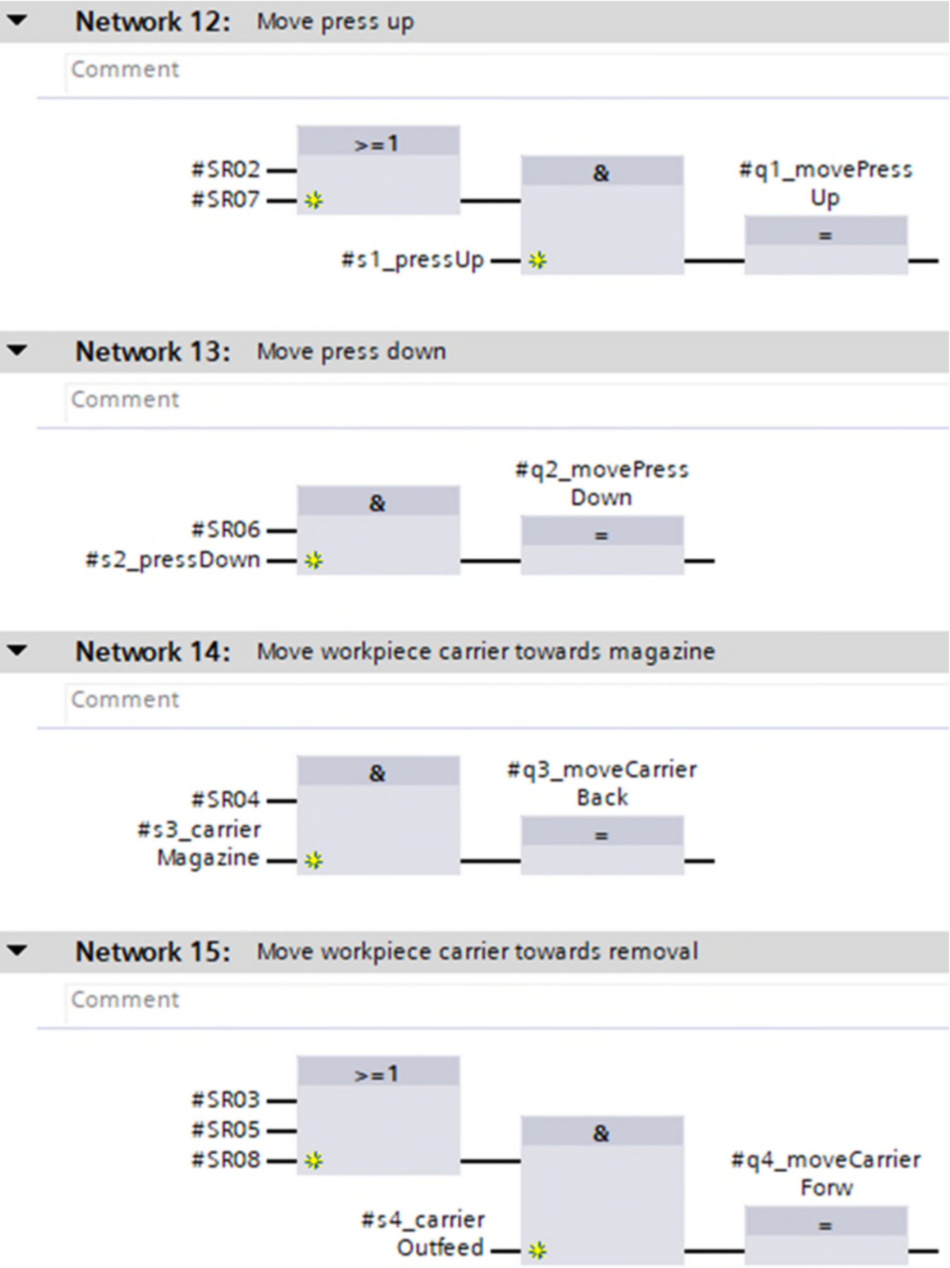


Network 5: Step 3

Comment

7. Assign the actions in the next 4 networks below the step chain.

It must be ensured that the movements of the motors are only controlled until the corresponding limit switches are reached, as otherwise they may run on block and be overloaded.
The control should therefore be a continuous action with a condition:



Network 12: Move press up

Comment

Network 13: Move press down

Comment

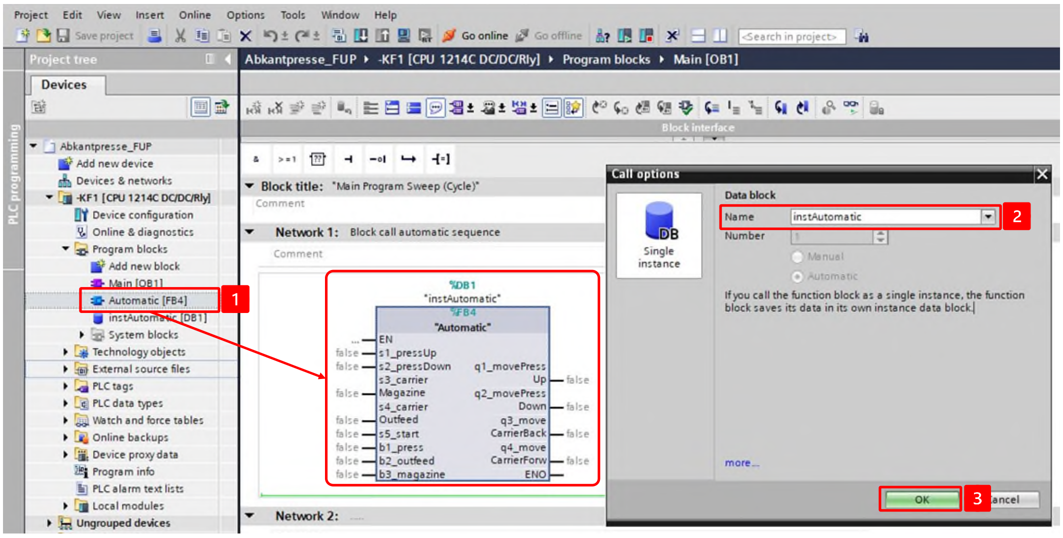
Network 14: Move workpiece carrier towards magazine

Comment

Network 15: Move workpiece carrier towards removal

Comment

8. Call up the function module in "MAIN" and create an instance:



Call options

Data block

Name: instAutomatic

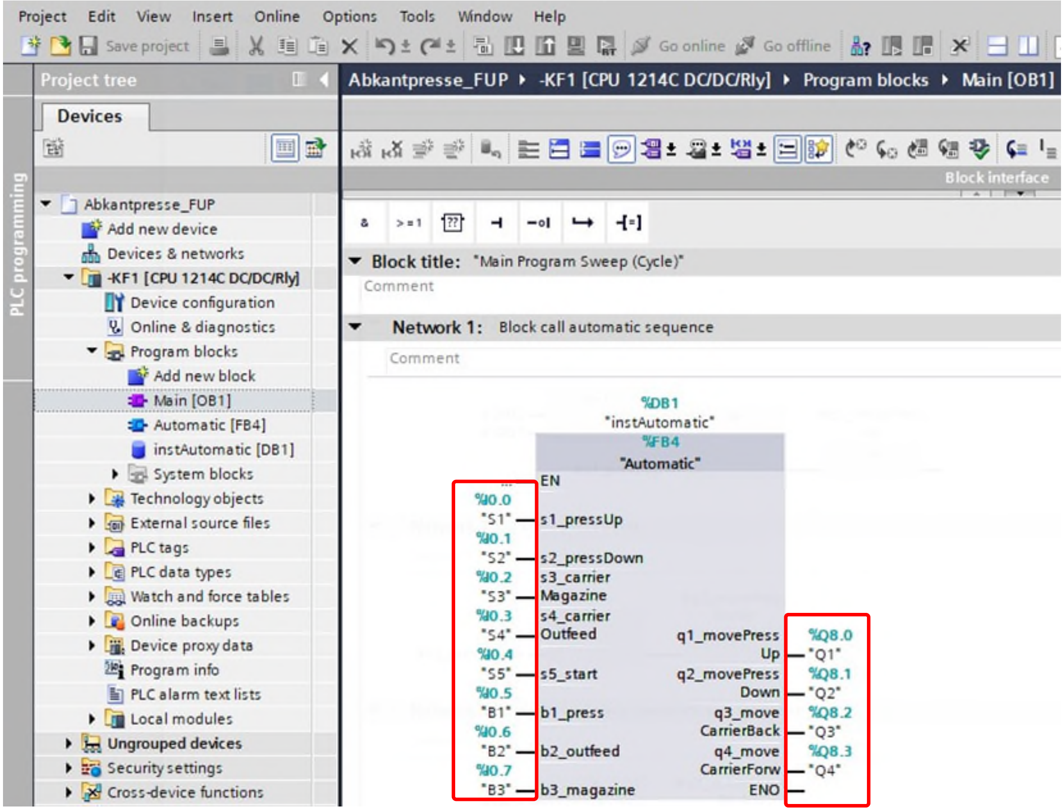
Number: 1

○ Manual

● Automatic

If you call the function block as a single instance, the function block saves its data in its own instance data block.

9. Connect the function block interface with the input and output variables from your variable table:



Input	Parameter	Parameter	Output
%I0.0 "S1"	s1_pressUp	q1_movePressUp	%Q8.0 "Q1"
%I0.1 "S2"	s2_pressDown	q2_movePressDown	%Q8.1 "Q2"
%I0.2 "S3"	s3_carrierMagazine	q3_moveCarrierBack	%Q8.2 "Q3"
%I0.3 "S4"	s4_carrierOutfeed	q4_moveCarrierForw	%Q8.3 "Q4"
%I0.4 "S5"	s5_start	ENO	
%I0.5 "B1"	b1_press		
%I0.6 "B2"	b2_outfeed		
%I0.7 "B3"	b3_magazine		

10. Commission the system in a structured manner with the aid of a commissioning protocol.

Solution

Solution:

The solution can be found in the TIA Portal project "Biegemaschine_FUP.zap17".

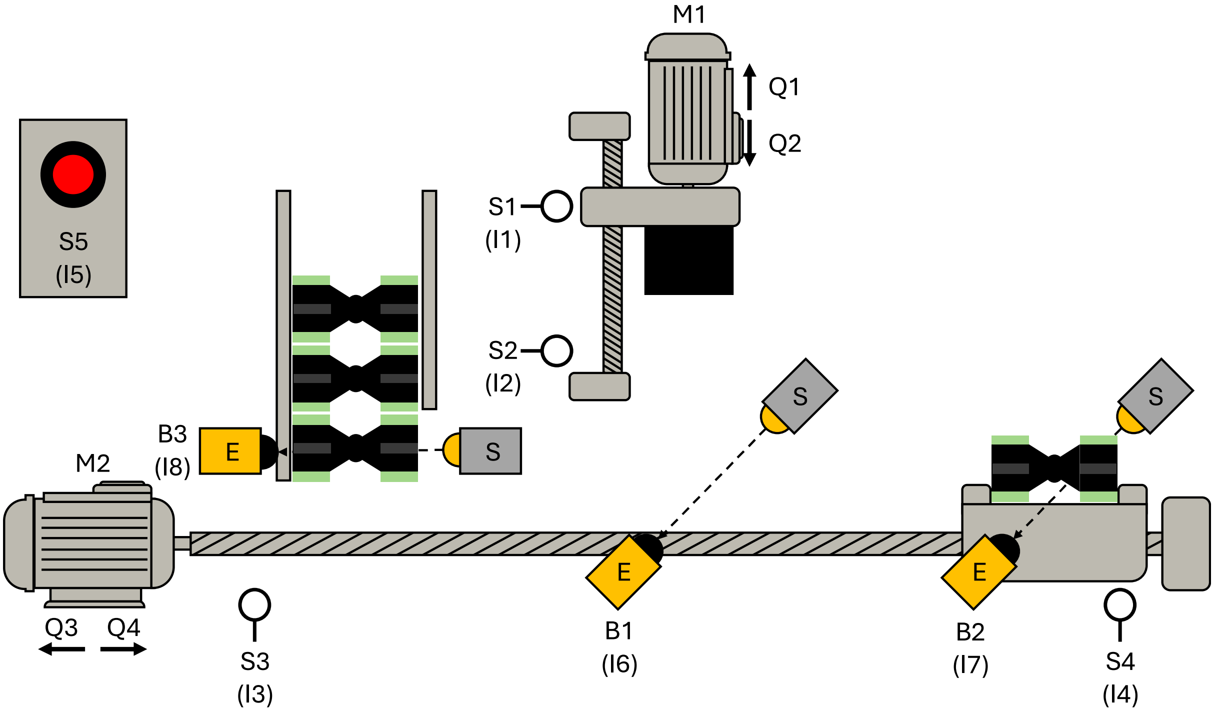
10.3 Exercise: Convert GRAFCET sequence chain into program code [ST / SCL].

Target:

I can create an automatic program based on the function description and the sequence chain created in GRAFCET.

Task:

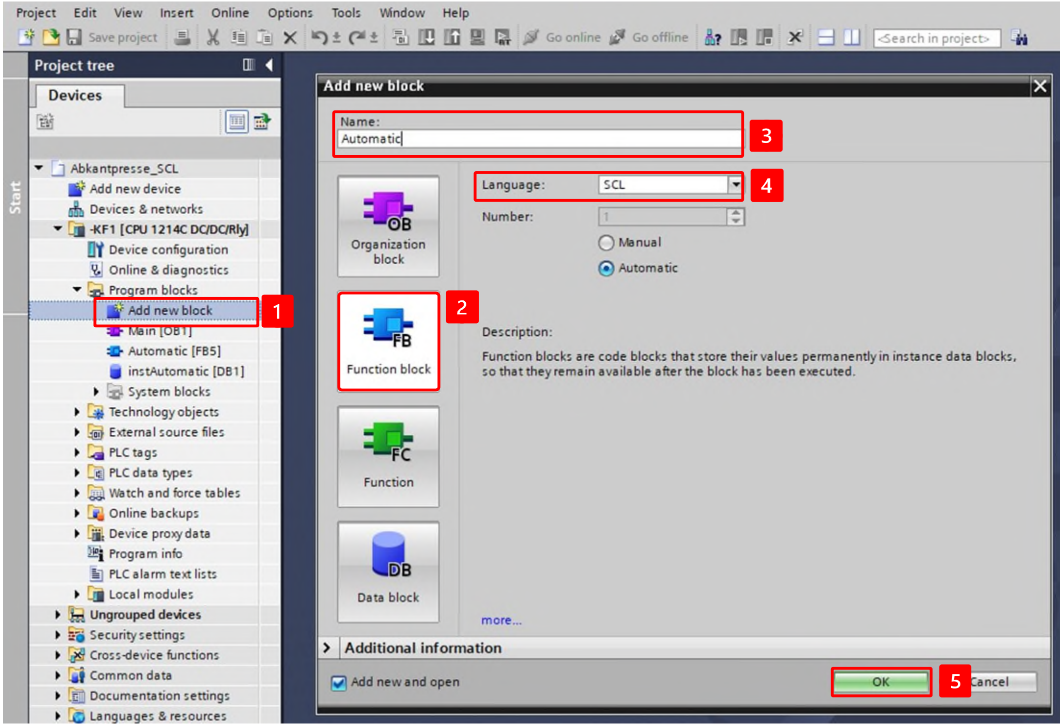
Create the PLC program for the automatic sequence for the bending machine so that the function is implemented based on the system description.



Picture 2 System diagram

Procedure:

1. Create a new function block, select the desired programming language and assign a meaningful name:



Name: Automatic

Language: SCL

Number: 1

○ Manual

● Automatic

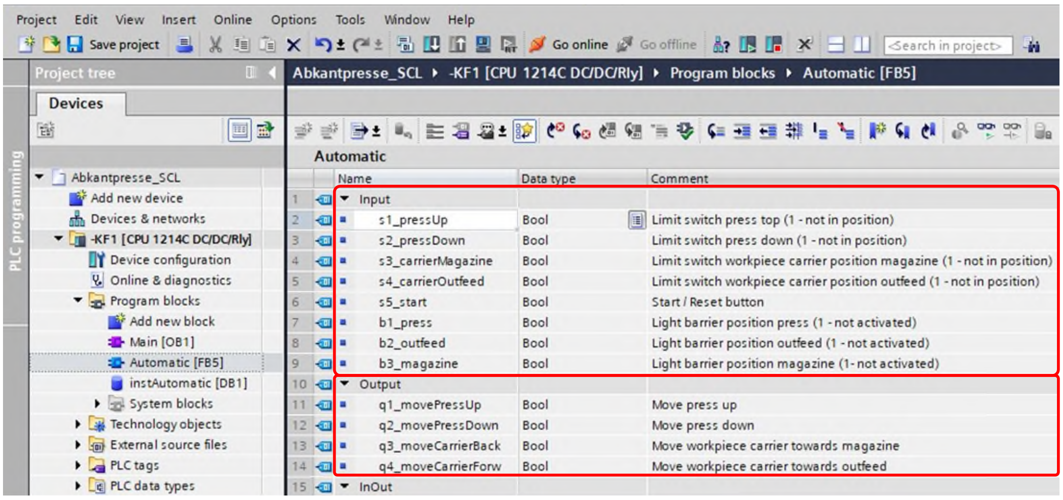
Description:

Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available after the block has been executed.

☑ Add new and open

OK Cancel

2. Declare variables for the sensors and actuators in the function block interface:

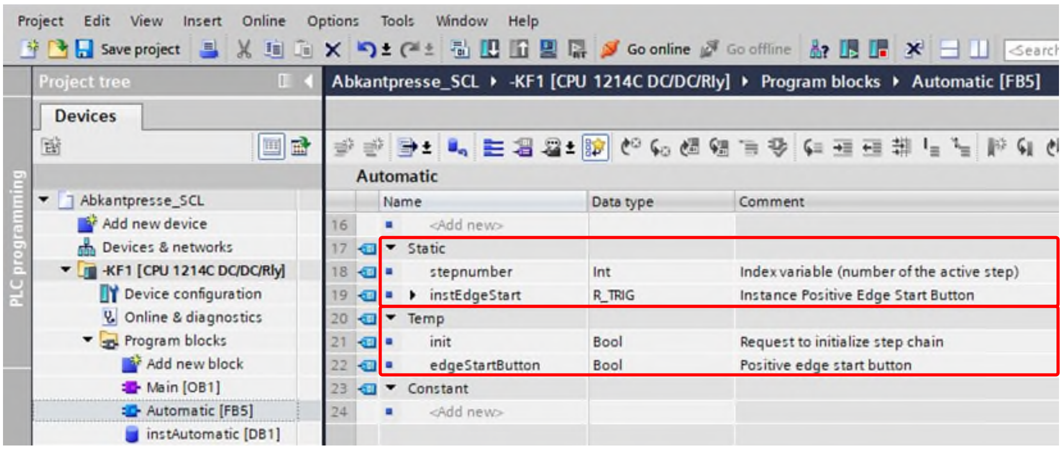


Abkantpresse_SCL ▸ -KF1 [CPU 1214C DC/DC/Rly] ▸ Program blocks ▸ Automatic [FB5]

Automatic

Name	Data type	Comment
▼ Input		
s1_pressUp	Bool	Limit switch press top (1 - not in position)
s2_pressDown	Bool	Limit switch press down (1 - not in position)
s3_carrierMagazine	Bool	Limit switch workpiece carrier position magazine (1 - not in position)
s4_carrierOutfeed	Bool	Limit switch workpiece carrier position outfeed (1 - not in position)
s5_start	Bool	Start / Reset button
b1_press	Bool	Light barrier position press (1 - not activated)
b2_outfeed	Bool	Light barrier position outfeed (1 - not activated)
b3_magazine	Bool	Light barrier position magazine (1- not activated)
▼ Output		
q1_movePressUp	Bool	Move press up
q2_movePressDown	Bool	Move press down
q3_moveCarrierBack	Bool	Move workpiece carrier towards magazine
q4_moveCarrierForw	Bool	Move workpiece carrier towards outfeed
▼ InOut		

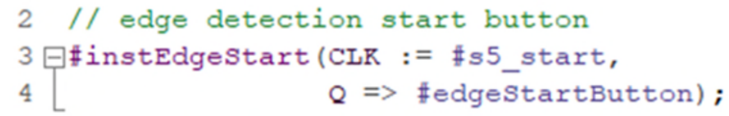
3. Declare the index variable in the static area of the function block interface, an instance for the edge evaluation of the start button, as well as a variable for initialization and for the positive edge of the start button in the temporary area:



Automatic

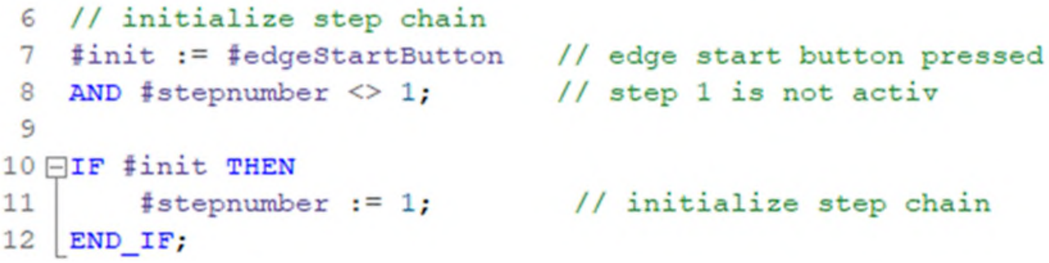
	Name	Data type	Comment
16	<Add new>		
17	▼ Static		
18	stepnumber	Int	Index variable (number of the active step)
19	▸ instEdgeStart	R_TRIG	Instance Positive Edge Start Button
20	▼ Temp		
21	init	Bool	Request to initialize step chain
22	edgeStartButton	Bool	Positive edge start button
23	▼ Constant		
24	<Add new>		

4. Program the evaluation of the positive edge of the start button (S5):



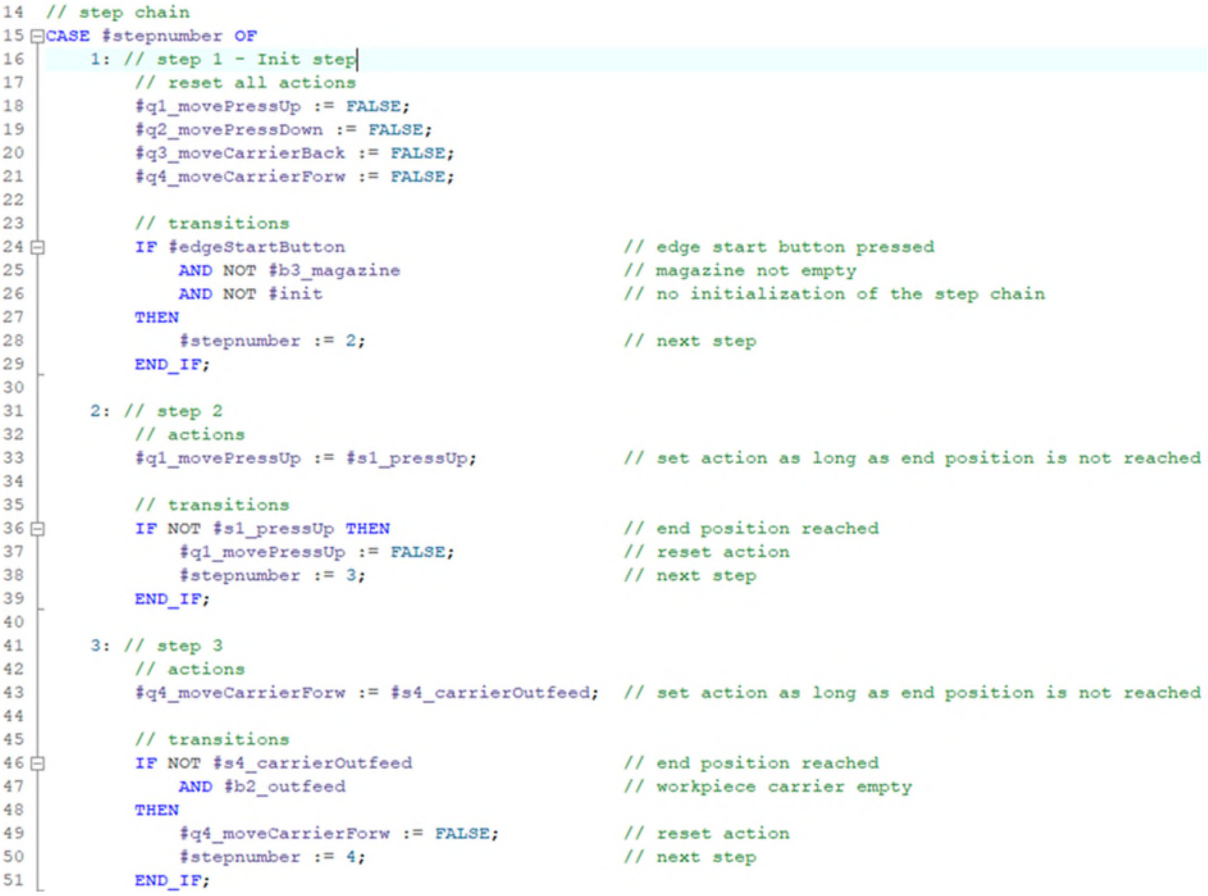
```
// edge detection start button
#instEdgeStart(CLK := #s5_start,
               Q => #edgeStartButton);
```

5. Then program the initialization of the chain. The function description indicates that the step chain should be initialized when the start button is pressed (positive edge) and the chain is not in the initial step:



```
// initialize step chain
#init := #edgeStartButton    // edge start button pressed
AND #stepnumber <> 1;        // step 1 is not activ

IF #init THEN
    #stepnumber := 1;         // initialize step chain
END_IF;
```

6. Implement the individual steps from the GRAFCET in the following CASE structure. For each step, create a new CASE in the structure that maps the step number:

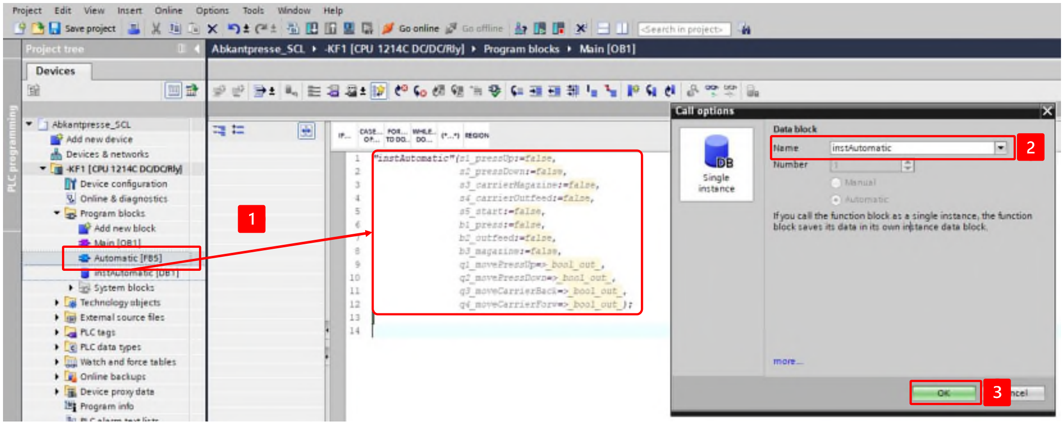


```
// step chain
CASE #stepnumber OF
    1: // step 1 - Init step
        // reset all actions
        #q1_movePressUp := FALSE;
        #q2_movePressDown := FALSE;
        #q3_moveCarrierBack := FALSE;
        #q4_moveCarrierForw := FALSE;

        // transitions
        IF #edgeStartButton                         // edge start button pressed
            AND NOT #b3_magazine                    // magazine not empty
            AND NOT #init                           // no initialization of the step chain
        THEN
            #stepnumber := 2;                       // next step
        END_IF;

    2: // step 2
        // actions
        #q1_movePressUp := #s1_pressUp;             // set action as long as end position is not reached

        // transitions
        IF NOT #s1_pressUp THEN                     // end position reached
            #q1_movePressUp := FALSE;               // reset action
            #stepnumber := 3;                       // next step
        END_IF;

    3: // step 3
        // actions
        #q4_moveCarrierForw := #s4_carrierOutfeed;  // set action as long as end position is not reached

        // transitions
        IF NOT #s4_carrierOutfeed                   // end position reached
            AND #b2_outfeed                         // workpiece carrier empty
        THEN
            #q4_moveCarrierForw := FALSE;           // reset action
            #stepnumber := 4;                       // next step
        END_IF;
```

All actions are reset in the initial step. This ensures that no actions remain set if an active step chain is aborted by an initialization request.

In the initial step, the transition is extended by "AND NOT Init", which prevents the step from being exited in the same cycle if an initialization request is made.

7. Call up the function module in "MAIN" and create an instance:



```
"instAutomatic"(s1_pressUp:=false,
                s2_pressDown:=false,
                s3_carrierMagazine:=false,
                s4_carrierOutfeed:=false,
                s5_start:=false,
                b1_press:=false,
                b2_outfeed:=false,
                b3_magazine:=false,
                q1_movePressUp=>_bool_out_,
                q2_movePressDown=>_bool_out_,
                q3_moveCarrierBack=>_bool_out_,
                q4_moveCarrierForw=>_bool_out_);
```

Data block

Name: instAutomatic

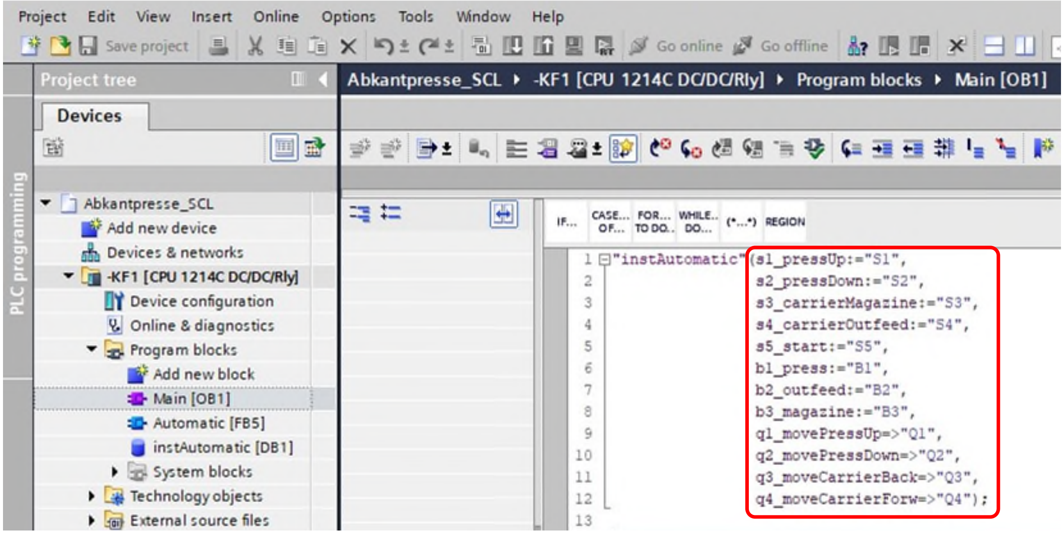
Number: 1

○ Manual

● Automatic

If you call the function block as a single instance, the function block saves its data in its own instance data block.

8. Connect the function block interface with the input and output variables from your variable table:



```
"instAutomatic"(s1_pressUp:="S1",
                s2_pressDown:="S2",
                s3_carrierMagazine:="S3",
                s4_carrierOutfeed:="S4",
                s5_start:="S5",
                b1_press:="B1",
                b2_outfeed:="B2",
                b3_magazine:="B3",
                q1_movePressUp=>"Q1",
                q2_movePressDown=>"Q2",
                q3_moveCarrierBack=>"Q3",
                q4_moveCarrierForw=>"Q4");
```

9. Commission the system in a structured manner with the aid of a commissioning protocol.

Solution

Solution:

The solution can be found in the TIA Portal project "Biegemaschine_SCL.zap17".

10.4 Exercise: Extending the sequence chain with GRAFCET [1] - Time function

Target:

I can extend an existing GRAFCET sequence chain.

Task:

Expand your existing GRAFCET sequence chain so that the following functional description is fulfilled.

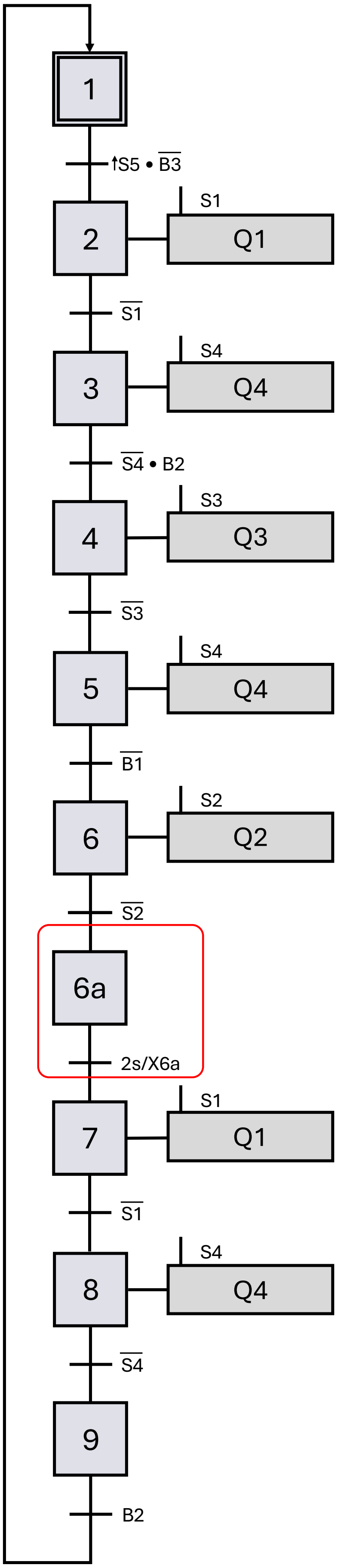
Function:

The bending process should be adjusted. The press is currently lowered (step 6) until the lower end position (S2 actuated) is reached. The press then moves back to its starting position (S1 actuated) (step 7).

In future, after the press has reached the lower end position, the system should not switch to step 7, but to a wait step (step 6a). This step must be activated for 2 seconds. After this time has elapsed, you can switch to step 7 and thus start the upward movement.

Solution

Solution:


1
↑S5 • $\overline{B3}$
2
S1
Q1
$\overline{S1}$
3
S4
Q4
$\overline{S4}$ • B2
4
S3
Q3
$\overline{S3}$
5
S4
Q4
$\overline{B1}$
6
S2
Q2
$\overline{S2}$
6a
2s/X6a
7
S1
Q1
$\overline{S1}$
8
S4
Q4
$\overline{S4}$
9
B2

10.5 Exercise: Converting a modified GRAFCET sequence chain into program code [FBD] - Time function

Target:

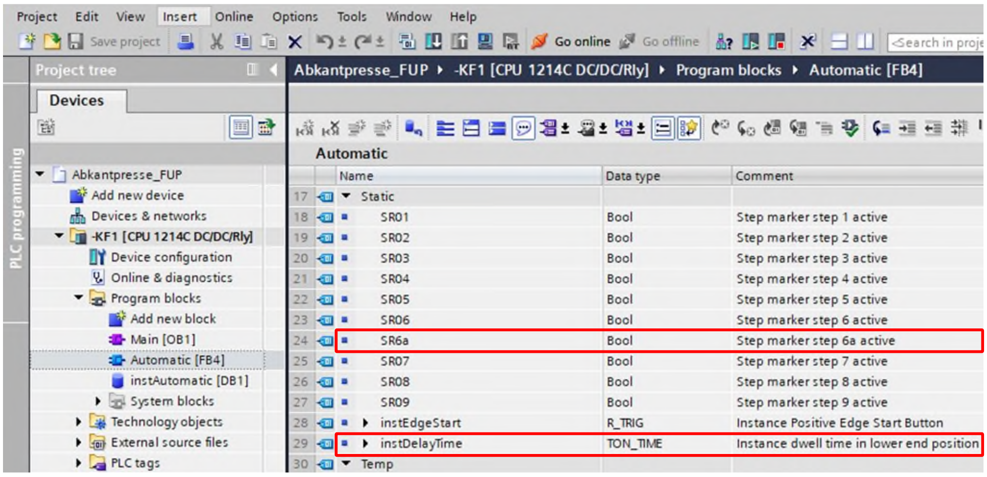
I can adapt an existing automatic program using the function description and the sequence chain changed in GRAFCET.

Task:

Adapt the PLC program for the automatic sequence for the bending machine so that the function is implemented using the modified system description from the exercise "Extend GRAFCET sequence chain [1] - time function".

Procedure:

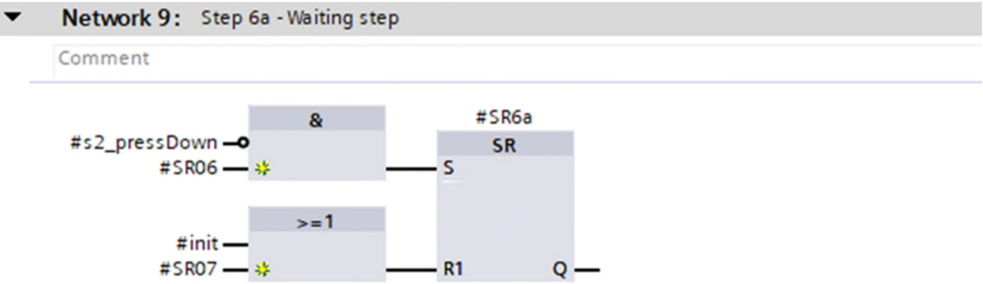
1. Declare a new step flag and an instance for the delay time in the static area of the function block interface:



Automatic

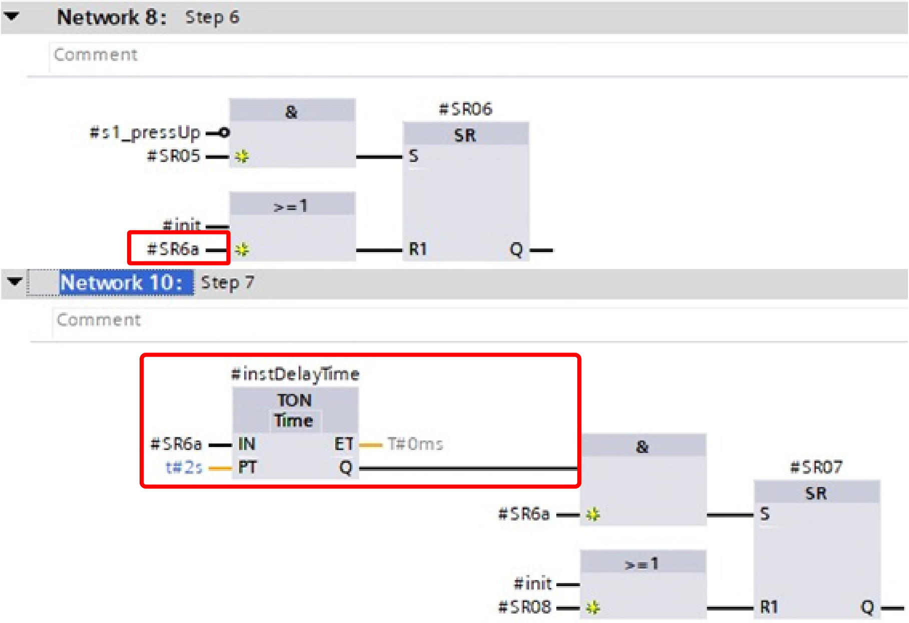
	Name	Data type	Comment
17	▼ Static		
18	SR01	Bool	Step marker step 1 active
19	SR02	Bool	Step marker step 2 active
20	SR03	Bool	Step marker step 3 active
21	SR04	Bool	Step marker step 4 active
22	SR05	Bool	Step marker step 5 active
23	SR06	Bool	Step marker step 6 active
24	SR6a	Bool	Step marker step 6a active
25	SR07	Bool	Step marker step 7 active
26	SR08	Bool	Step marker step 8 active
27	SR09	Bool	Step marker step 9 active
28	▸ instEdgeStart	R_TRIG	Instance Positive Edge Start Button
29	▸ instDelayTime	TON_TIME	Instance dwell time in lower end position
30	▼ Temp		

2. Program the new step using the SR element:


Network 9: Step 6a - Waiting step

Comment

3. Adjust the set and reset conditions of the previous and subsequent steps:


Network 8: Step 6

Comment

Network 10: Step 7

Comment

4. Commission the system in a structured manner with the aid of a commissioning protocol.

Solution

Solution:

The solution can be found in the TIA Portal project "Biegemaschine_Erw_1_Zeitfunktion_FUP.zap17".

10.6 Exercise: Converting a modified GRAFCET sequence chain into program code [ST / SCL] - Time function

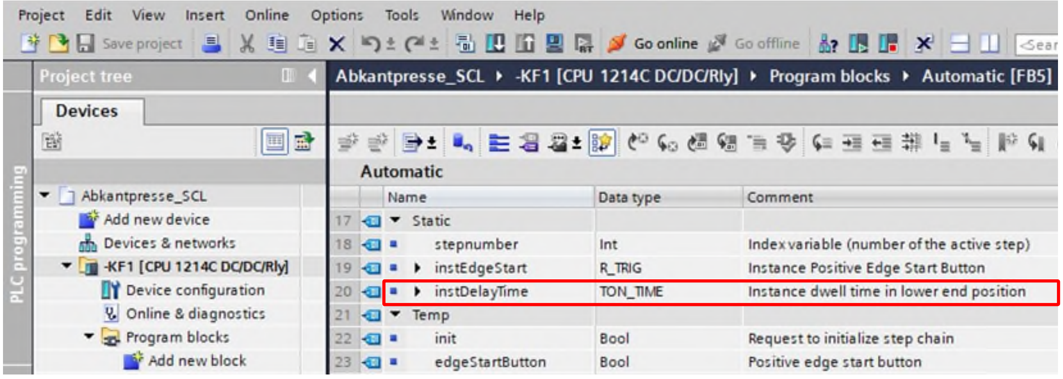
Target:

I can adapt an existing automatic program using the function description and the sequence chain changed in GRAFCET.

Task:

Adapt the PLC program for the automatic sequence for the bending machine so that the function is implemented using the modified system description from the exercise "Extend GRAFCET sequence chain [1] - time function".

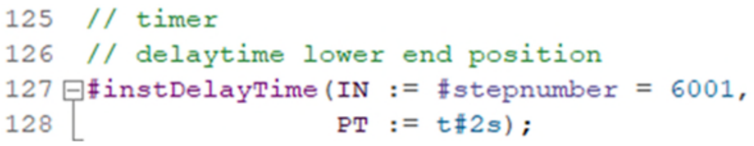
1. Declare an instance for the delay time in the static area of the function block interface:



Automatic

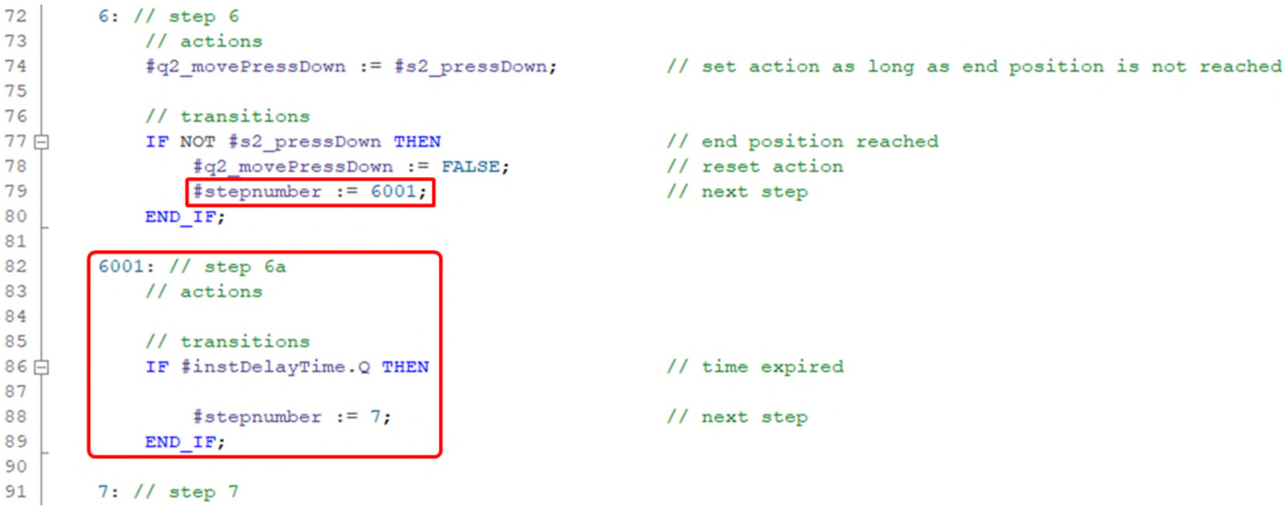
	Name	Data type	Comment
17	▾ Static		
18	stepnumber	Int	Index variable (number of the active step)
19	▸ instEdgeStart	R_TRIG	Instance Positive Edge Start Button
20	▸ instDelayTime	TON_TIME	Instance dwell time in lower end position
21	▾ Temp		
22	init	Bool	Request to initialize step chain
23	edgeStartButton	Bool	Positive edge start button

2. Program the wait time below the CASE structure. A free number must be used as the index for the step, for example 6001:



```
// timer
// delaytime lower end position
#instDelayTime(IN := #stepnumber = 6001,
               PT := t#2s);
```

3. Program the step:



```
6: // step 6
    // actions
    #q2_movePressDown := #s2_pressDown;          // set action as long as end position is not reached

    // transitions
    IF NOT #s2_pressDown THEN                    // end position reached
        #q2_movePressDown := FALSE;              // reset action
        #stepnumber := 6001;                     // next step
    END_IF;

6001: // step 6a
    // actions

    // transitions
    IF #instDelayTime.Q THEN                     // time expired

        #stepnumber := 7;                        // next step
    END_IF;

7: // step 7
```

4. Commission the system in a structured manner with the aid of a commissioning protocol.

Solution

Solution:

The solution can be found in the TIA Portal project "Biegemaschine_Erw_1_Zeitfunktion_SCL.zap17".

10.7 Exercise: Extending the sequence chain with GRAFCET [2] - Counting function

Target:

I can extend an existing GRAFCET sequence chain.

Task:

Extend your existing GRAFCET sequence chain from exercise "Extending the sequence chain with GRAFCET [1] - Time function" so that the following function description is fulfilled.

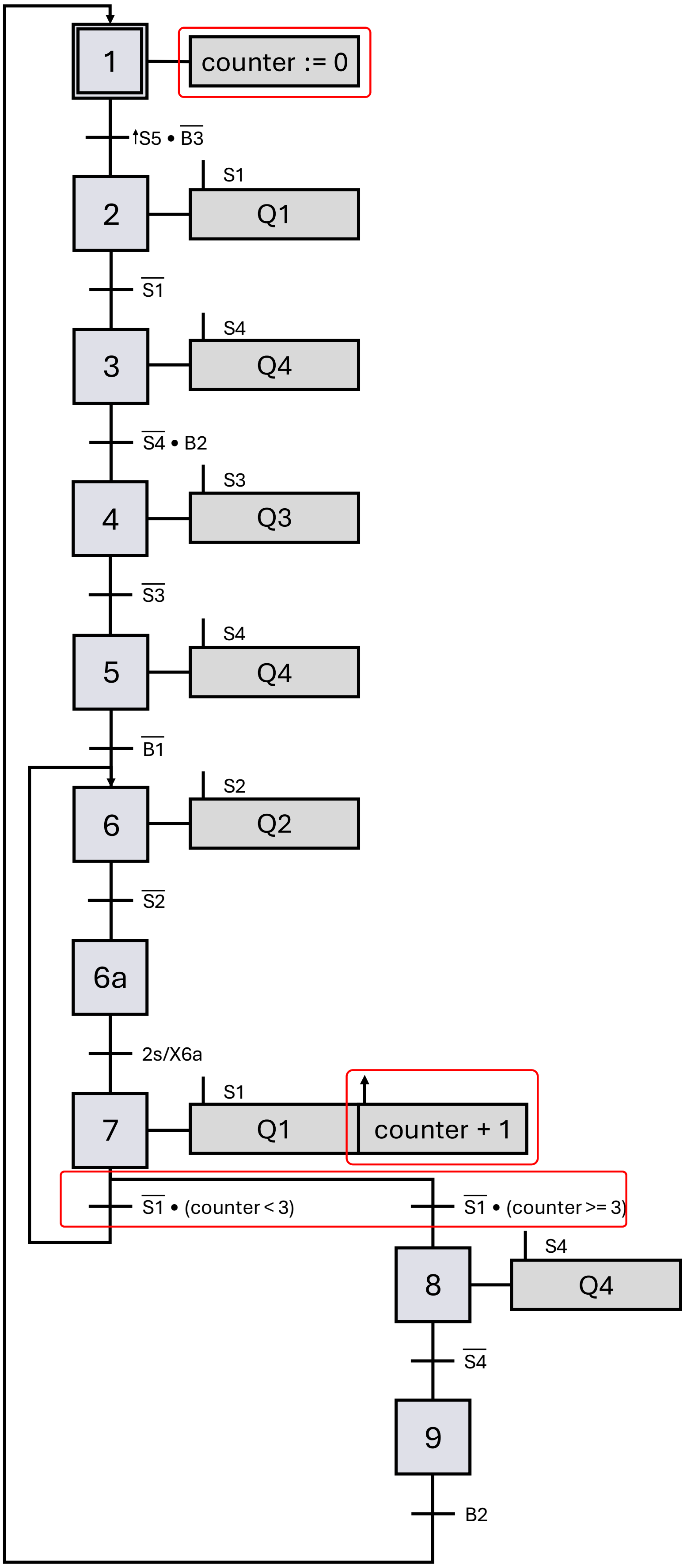
Function:

The bending process should be adjusted. The press currently remains in its lower end position for some time (step 6a) before the upward movement is started.
This already has a positive effect on the workpiece quality.
In addition to the waiting time, the bending process should now be carried out three times in succession for each workpiece.

The step chain must be adjusted so that the bending processes are counted when the press is opened. If 3 processing operations have not yet been recorded, the bending process must be restarted.
Once the workpiece has been bent three times, it can be moved on for acceptance.

Solution

Solution:


1
counter := 0
↑S5 • $\overline{B3}$
2
S1
Q1
$\overline{S1}$
3
S4
Q4
$\overline{S4}$ • B2
4
S3
Q3
$\overline{S3}$
5
S4
Q4
$\overline{B1}$
6
S2
Q2
$\overline{S2}$
6a
2s/X6a
7
S1
Q1
counter + 1
$\overline{S1}$ • (counter < 3)
$\overline{S1}$ • (counter >= 3)
8
S4
Q4
$\overline{S4}$
9
B2

10.8 Exercise: Converting a modified GRAFCET sequence chain into program code [FBD] - Counting function

Target:

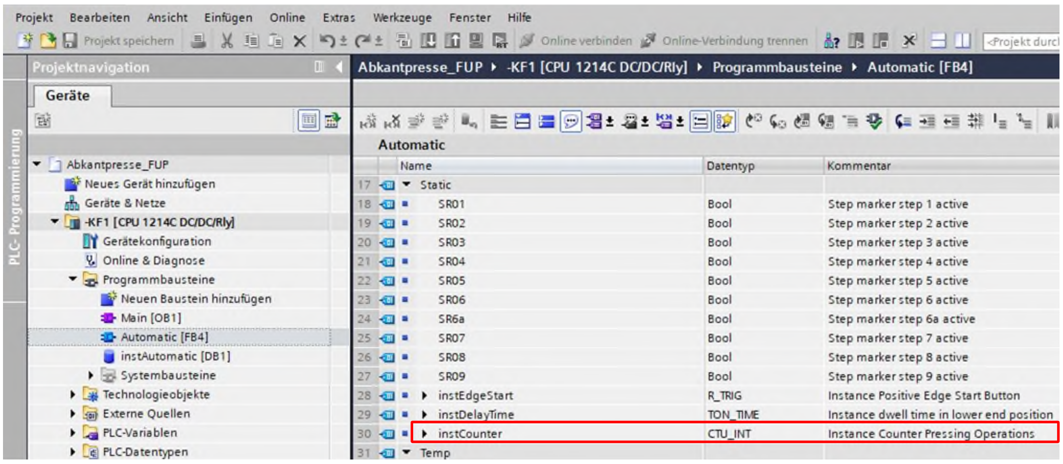
I can adapt an existing automatic program using the function description and the sequence chain changed in GRAFCET.

Task:

Adapt the PLC program for the automatic sequence for the bending machine so that the function is implemented using the modified system description from the exercise "Extending the GRAFCET sequence chain [2] - Counting function".

Procedure:

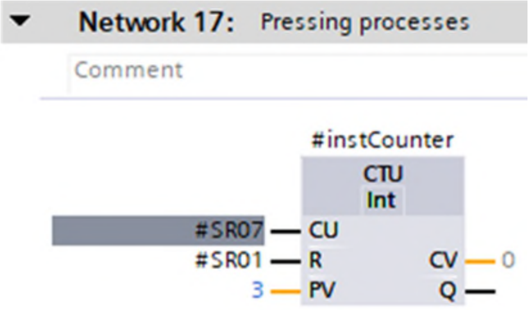
1. Declare an instance for the counter in the static area of the function block interface:



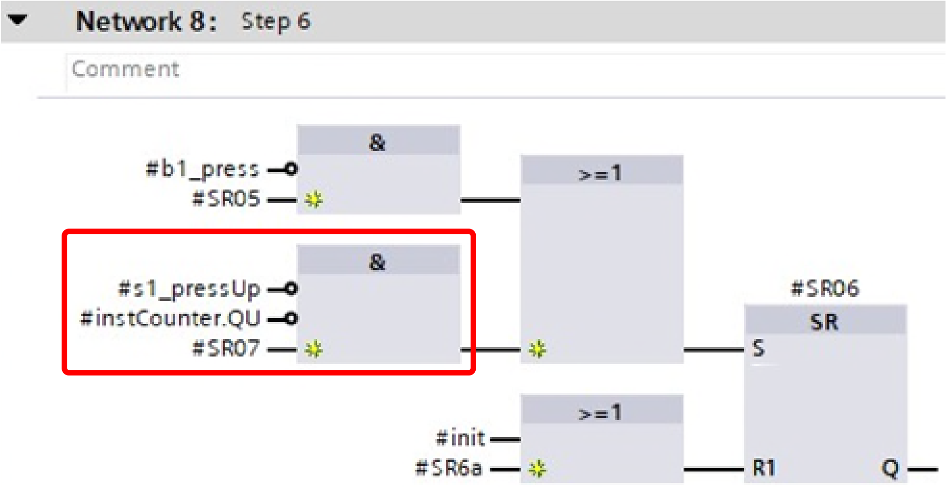
Automatic

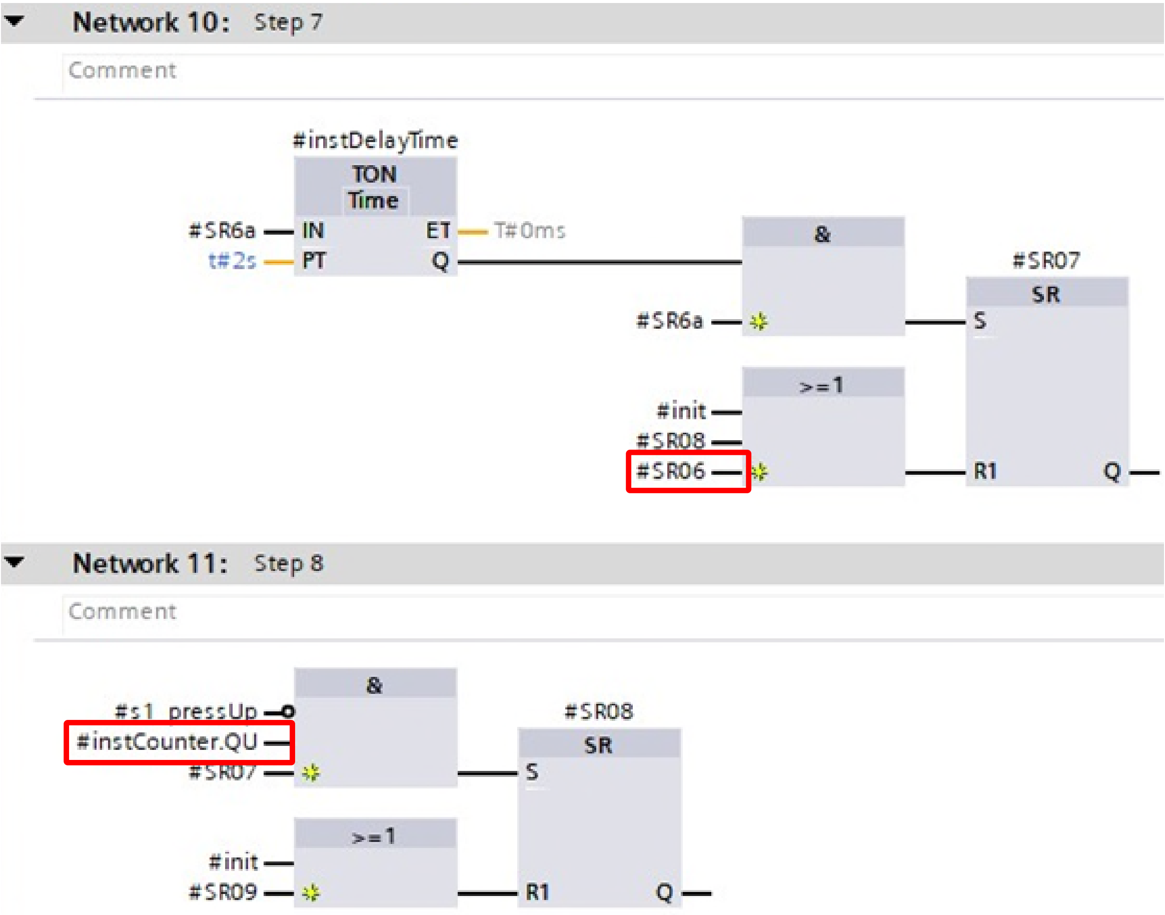
	Name	Datentyp	Kommentar
17	Static		
18	SR01	Bool	Step marker step 1 active
19	SR02	Bool	Step marker step 2 active
20	SR03	Bool	Step marker step 3 active
21	SR04	Bool	Step marker step 4 active
22	SR05	Bool	Step marker step 5 active
23	SR06	Bool	Step marker step 6 active
24	SR6a	Bool	Step marker step 6a active
25	SR07	Bool	Step marker step 7 active
26	SR08	Bool	Step marker step 8 active
27	SR09	Bool	Step marker step 9 active
28	instEdgeStart	R_TRIG	Instance Positive Edge Start Button
29	instDelayTime	TON_TIME	Instance dwell time in lower end position
30	instCounter	CTU_INT	Instance Counter Pressing Operations
31	Temp		

2. Program the meter in a new network:



3. Adjust the step chain:





4. Commission the system in a structured manner with the aid of a commissioning protocol.

Solution

Solution:

The solution can be found in the TIA Portal project "Biegemaschine_Erw_2_Zaehlfunktion_FUP.zap17".

10.9 Exercise: Converting a modified GRAFCET sequence chain into program code [ST / SCL].

Target:

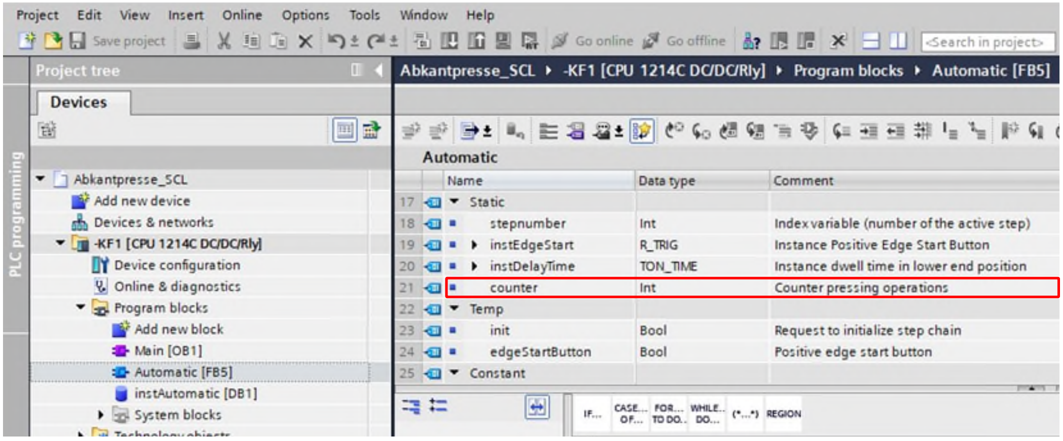
I can adapt an existing automatic program using the function description and the sequence chain changed in GRAFCET.

Task:

Adapt the PLC program for the automatic sequence for the bending machine so that the function is implemented using the modified system description from the exercise "Extending the GRAFCET sequence chain [2] - Counting function".

Procedure:

1. Declare a variable for the counter in the static area of the function block interface:

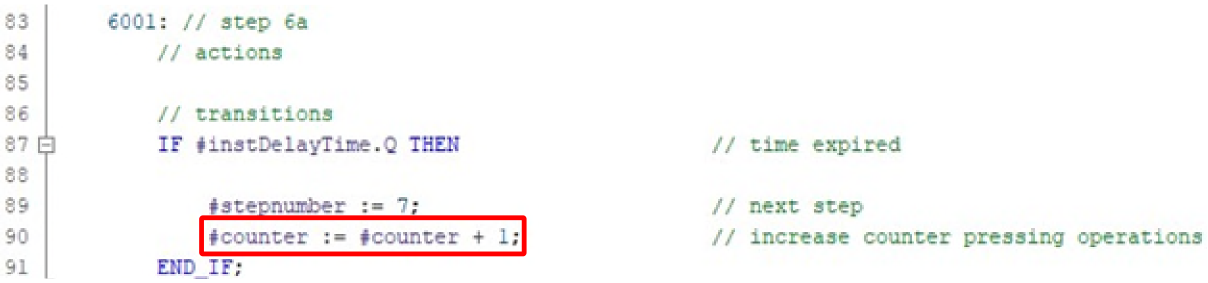


Abkantpresse_SCL ▸ -KF1 [CPU 1214C DC/DC/Rly] ▸ Program blocks ▸ Automatic [FB5]

Automatic

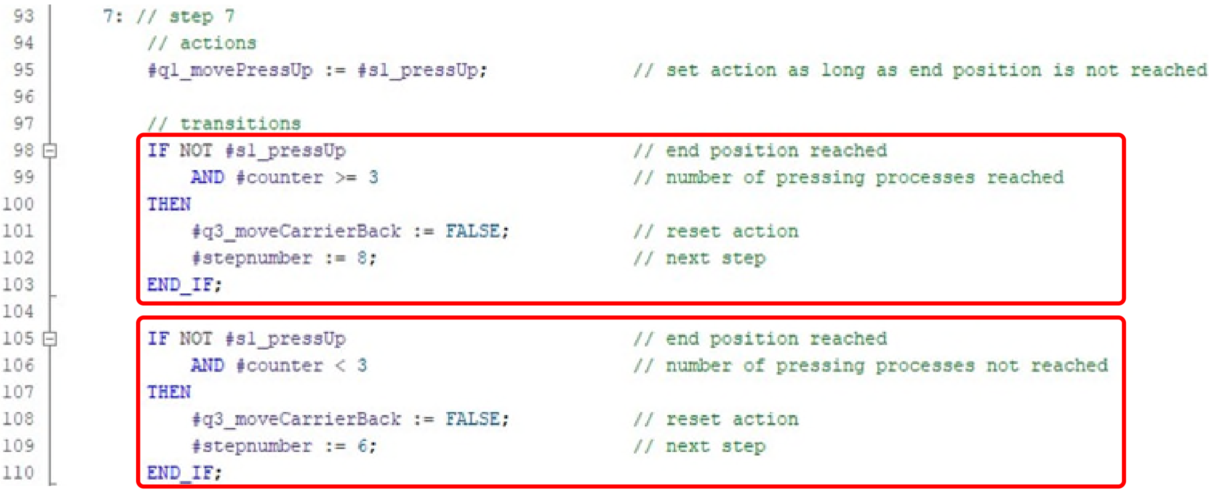
	Name	Data type	Comment
17	▾ Static		
18	stepnumber	Int	Index variable (number of the active step)
19	▸ instEdgeStart	R_TRIG	Instance Positive Edge Start Button
20	▸ instDelayTime	TON_TIME	Instance dwell time in lower end position
21	counter	Int	Counter pressing operations
22	▾ Temp		
23	init	Bool	Request to initialize step chain
24	edgeStartButton	Bool	Positive edge start button
25	▾ Constant		

2. Increase the counter when you jump to step 7 and reset the counter in the initial step:



```
6001: // step 6a
    // actions

    // transitions
    IF #instDelayTime.Q THEN                       // time expired

        #stepnumber := 7;                          // next step
        #counter := #counter + 1;                  // increase counter pressing operations
    END_IF;
```

3. Program the alternative branch:



```
7: // step 7
    // actions
    #q1_movePressUp := #s1_pressUp;            // set action as long as end position is not reached

    // transitions
    IF NOT #s1_pressUp                         // end position reached
        AND #counter >= 3                      // number of pressing processes reached
    THEN
        #q3_moveCarrierBack := FALSE;          // reset action
        #stepnumber := 8;                      // next step
    END_IF;

    IF NOT #s1_pressUp                         // end position reached
        AND #counter < 3                       // number of pressing processes not reached
    THEN
        #q3_moveCarrierBack := FALSE;          // reset action
        #stepnumber := 6;                      // next step
    END_IF;
```

4. Commission the system in a structured manner with the aid of a commissioning protocol.

Solution

Solution:

The solution can be found in the TIA Portal project "Biegemaschine_Erw_2_Zaehlfunktion_SCL.zap17".