Programação SFC com Factory IO

Henrique Silva Coutinho

Vantagens e desvantagens do SFC em relação ao Ladder

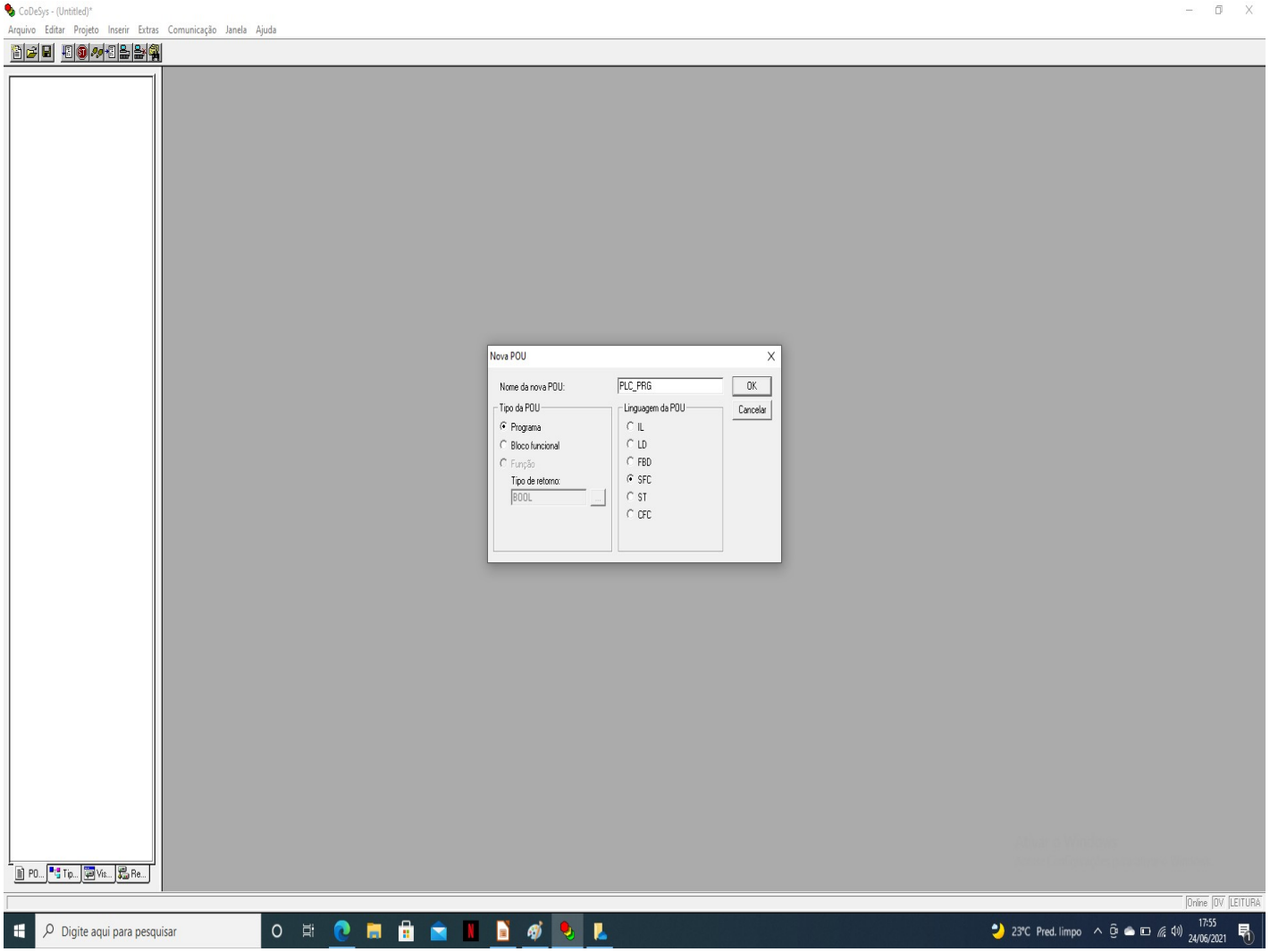
Vantagens:

- Programação sequencial
- Sincronismo
- Blocos que permitem programação diversa
- Paralelismo de processamento

Desvantagens:

- **Não possui algumas funções do Ladder**
- **Menos simples que o Ladder**
- **Intertravamento mais complexo que o Ladder**

Inicialiazando o SFC



Ao Criarmos um novo projeto selecionamos a linguagem SFC da POU.

Ou podemos acrescentar um objeto ao nosso projeto já iniciado

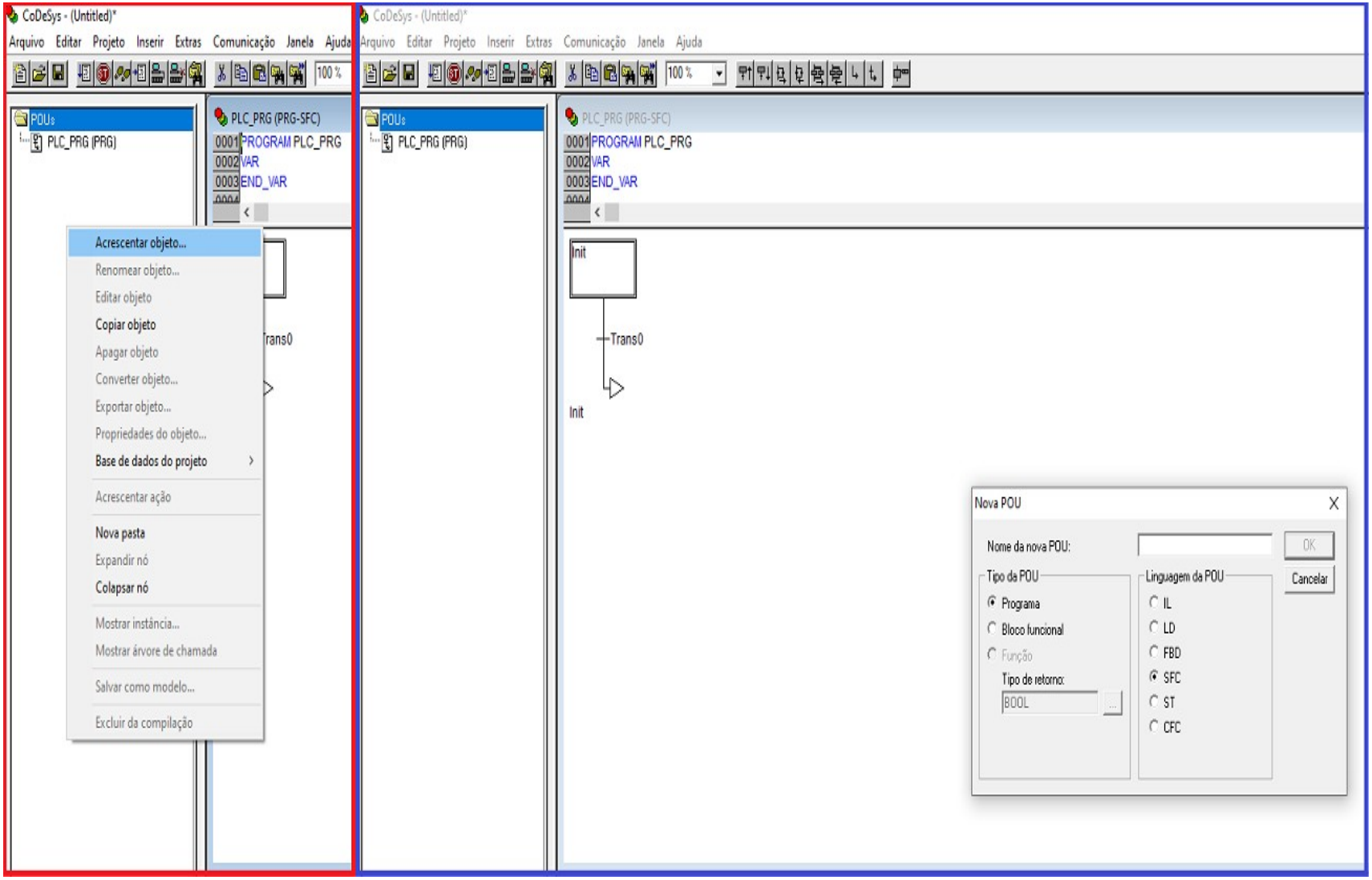
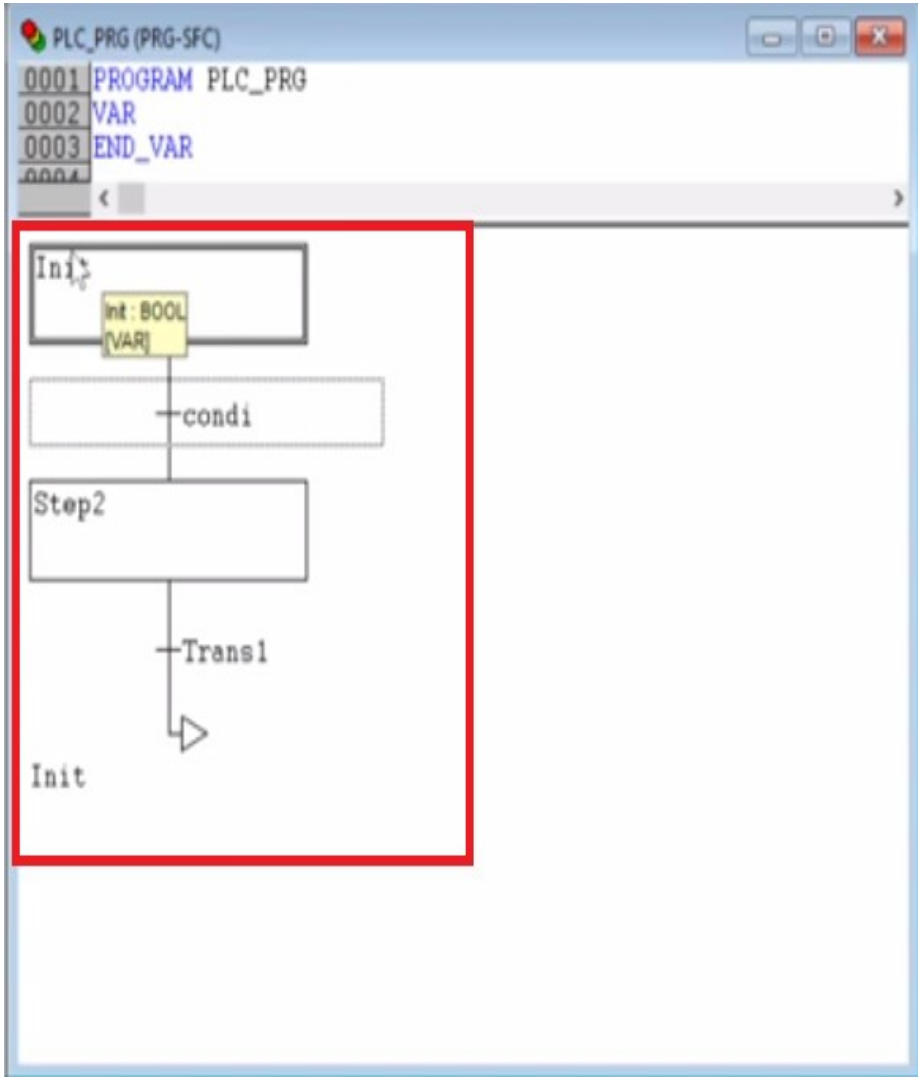
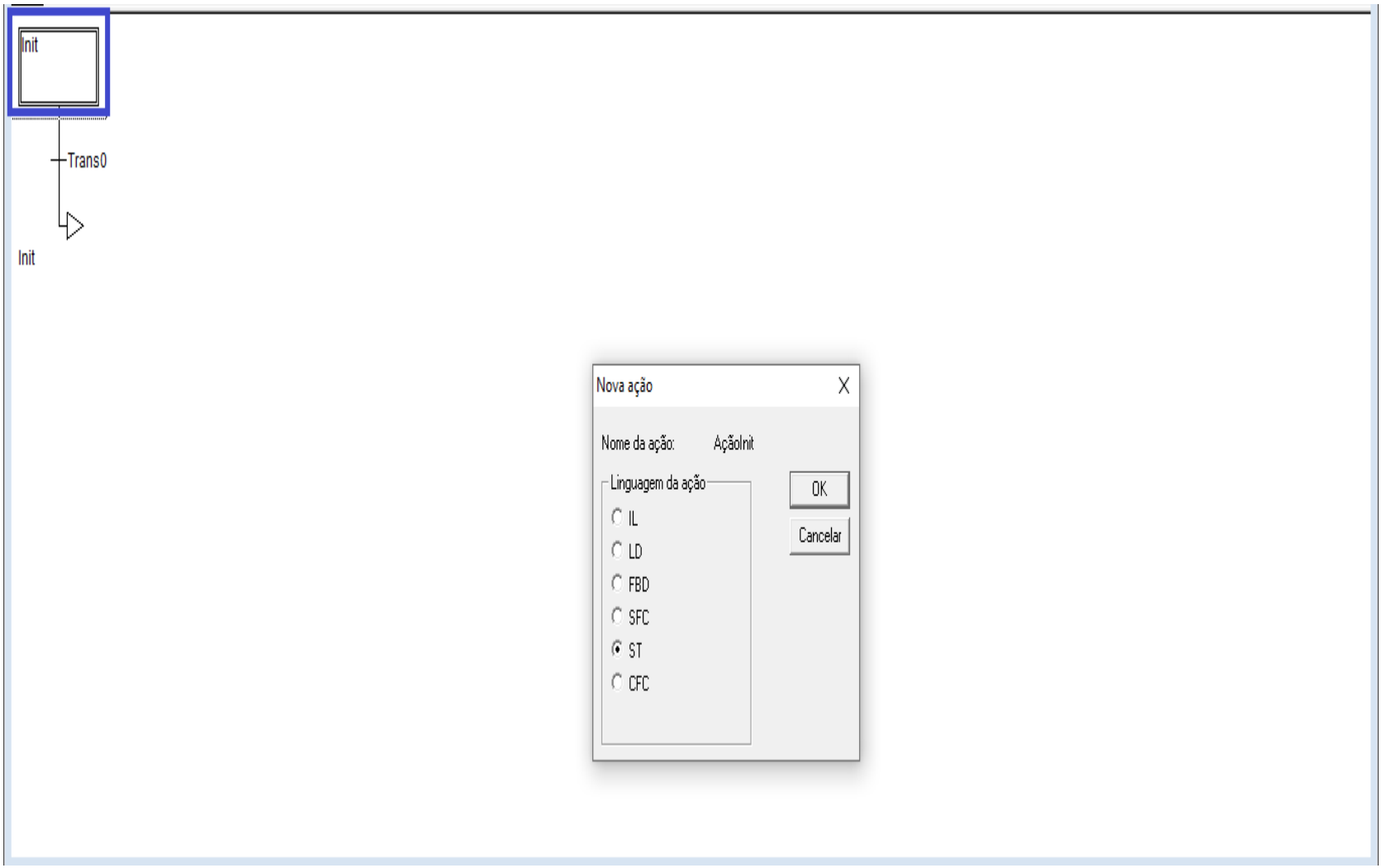


Diagrama SFC



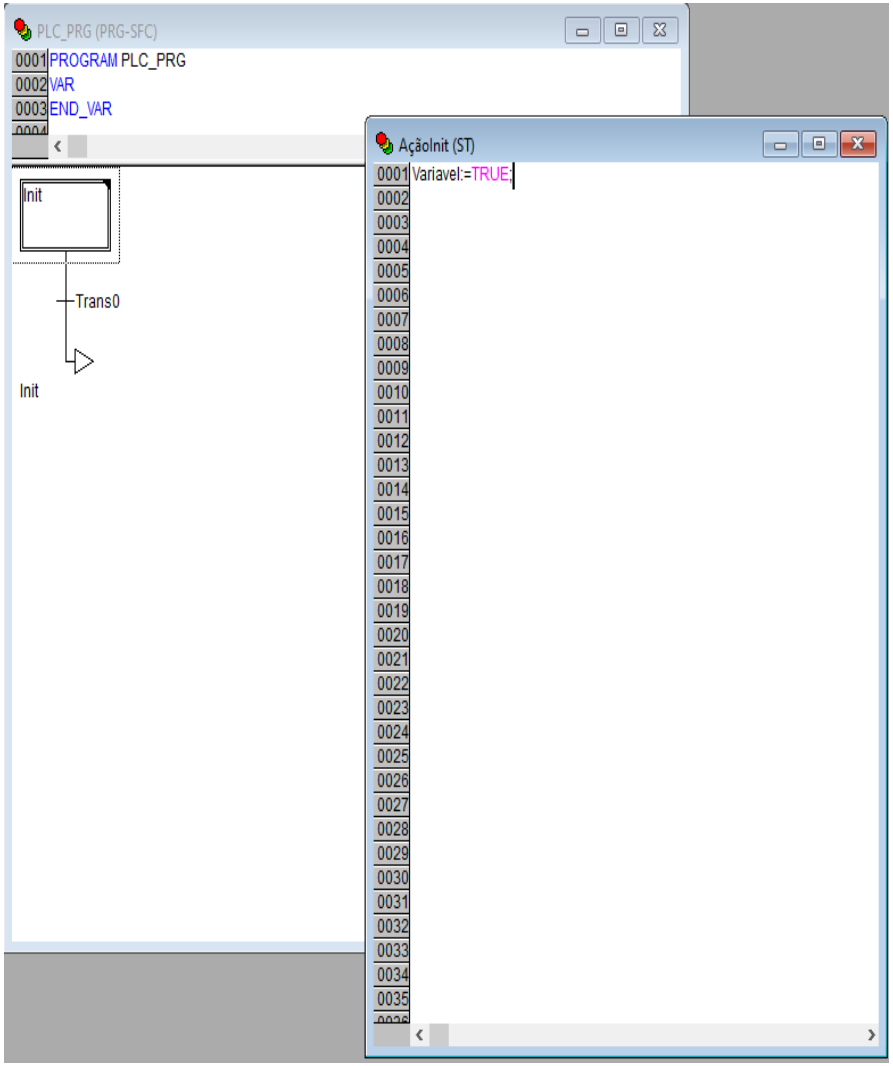
Um diagrama SFC possui o aspecto ao lado em vermelho, no qual há uma caixa que mostra o estado do sistema e dentro delas podemos inserir um programa.

Programando



Clicando no bloco em azul podemos começar a programar

Programando

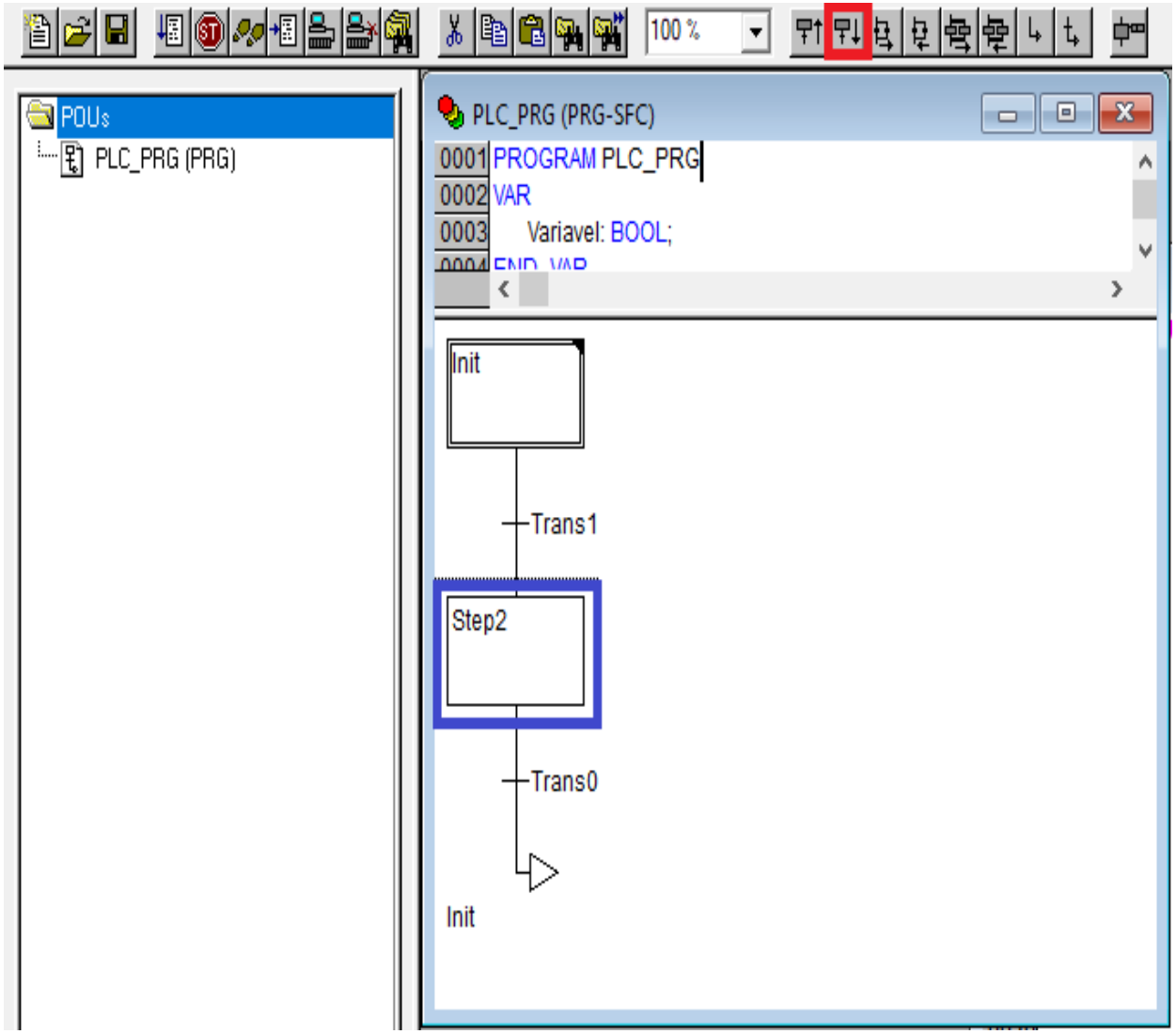


Abrirá uma janela para programarmos com a linguagem selecionada (No nosso caso ST).

Podemos então interagir com o diagrama, nesse caso pegamos uma variável e colocamos seu estado para TRUE.

Podemos programar em diversas linguagens.

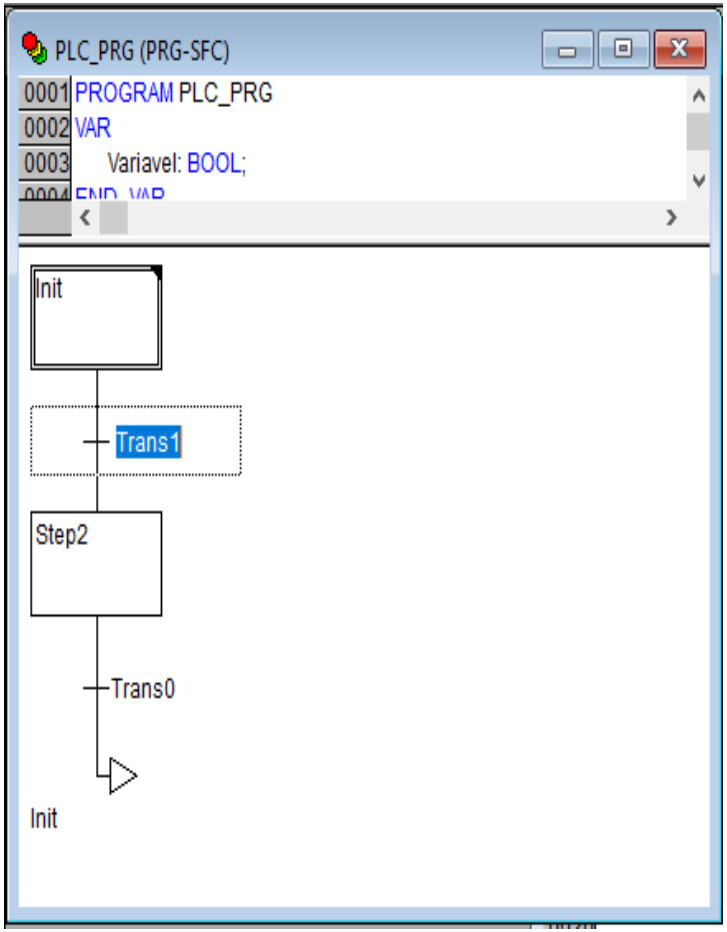
Adicionando outro bloco e programação



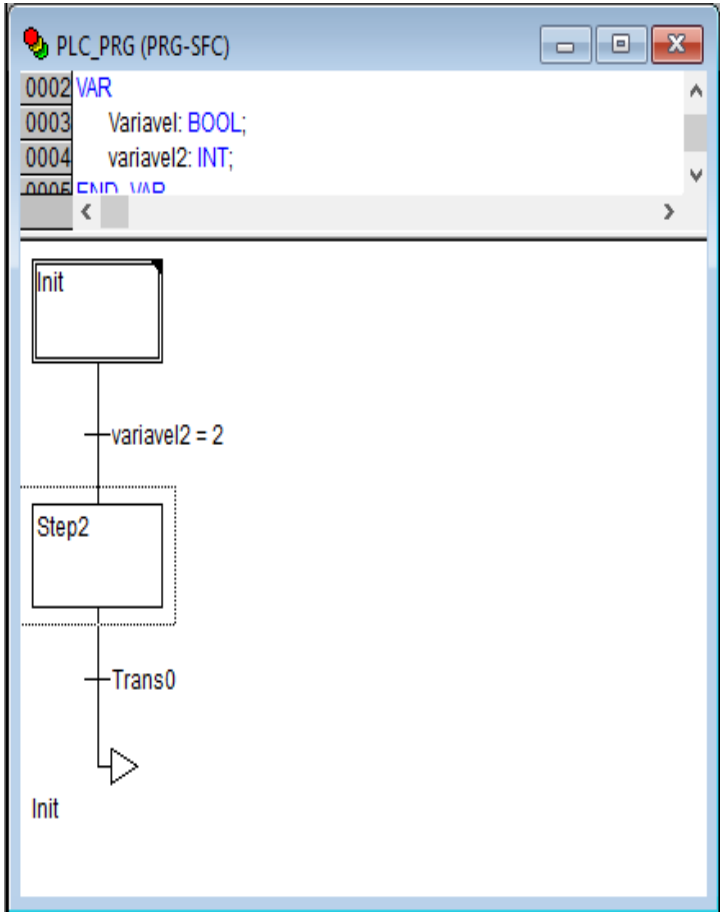
- Ao clicarmos no ícone em vermelho adicionamos um novo bloco de programação em azul.
- Observe que entre o primeiro e o segundo bloco há uma transição.
- A programação do segundo bloco só começará á operar quando a condição de transição for satisfeita.

O primeiro bloco é a inicialização, o segundo, programação

Alterando a transição

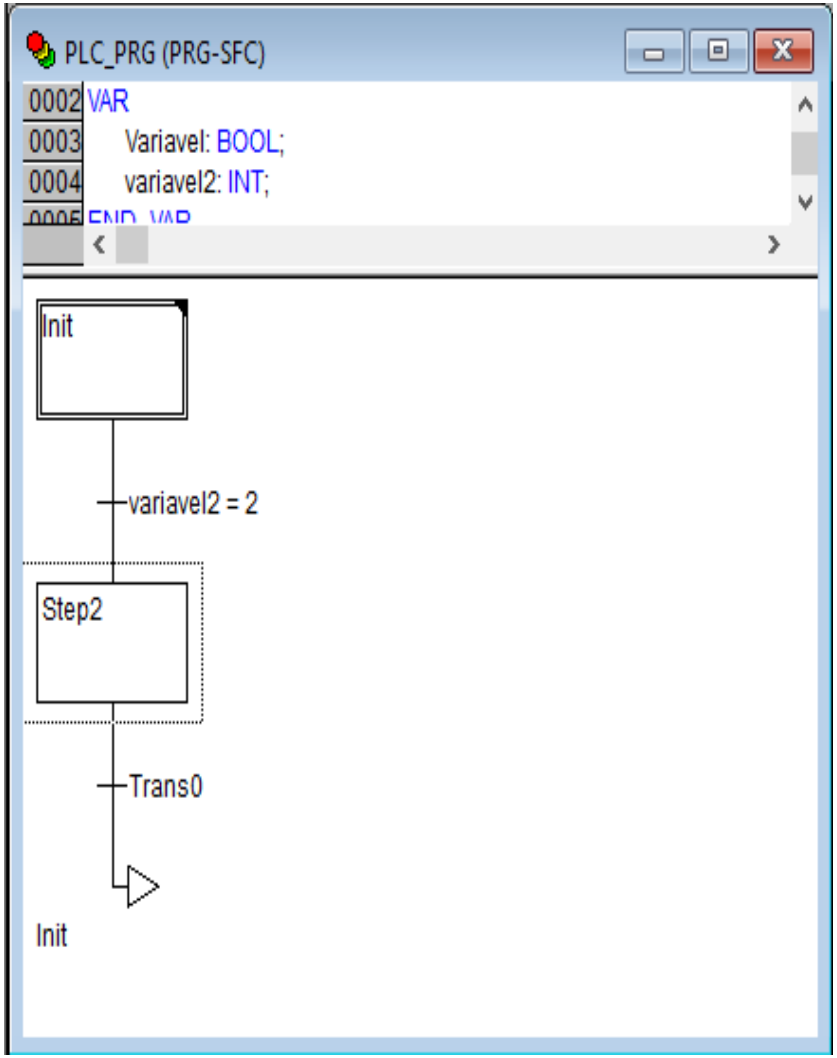


Inserimos uma condição para a passagem do bloco 1 (init) e o bloco 2



A variável2 foi declarada como INT

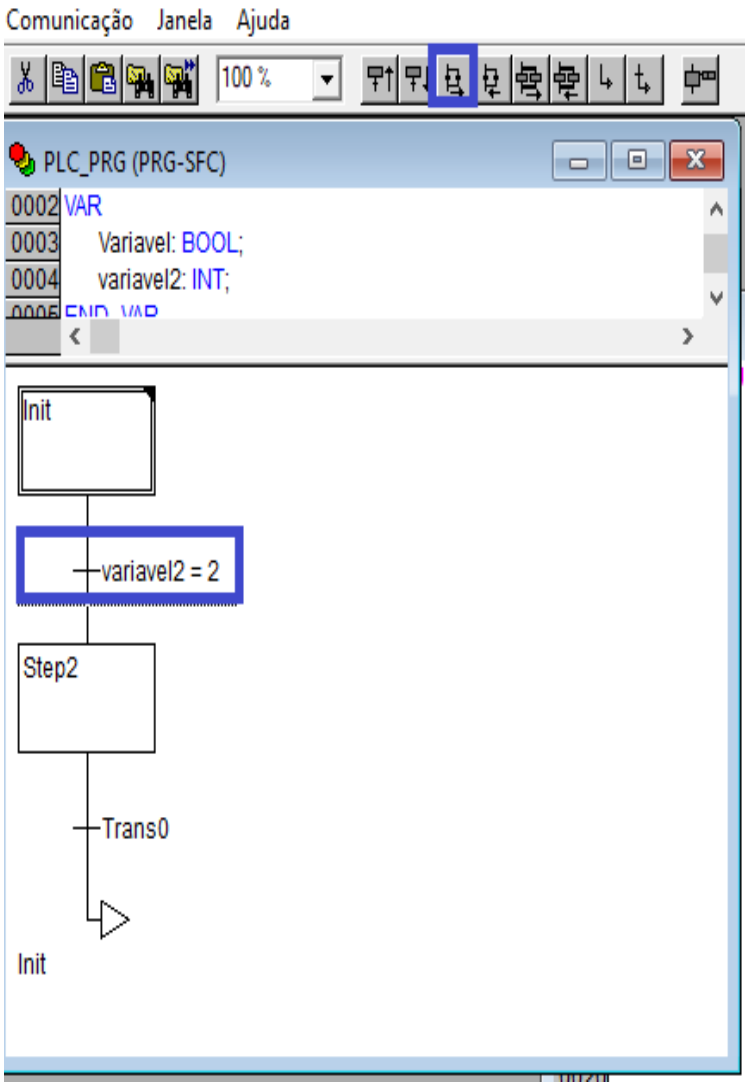
Repetição do programa



Abaixo do ultimo bloco há uma ultima transição que após ser cumprida faz com que o programa retorne ao bloco INIT e repita o processo.

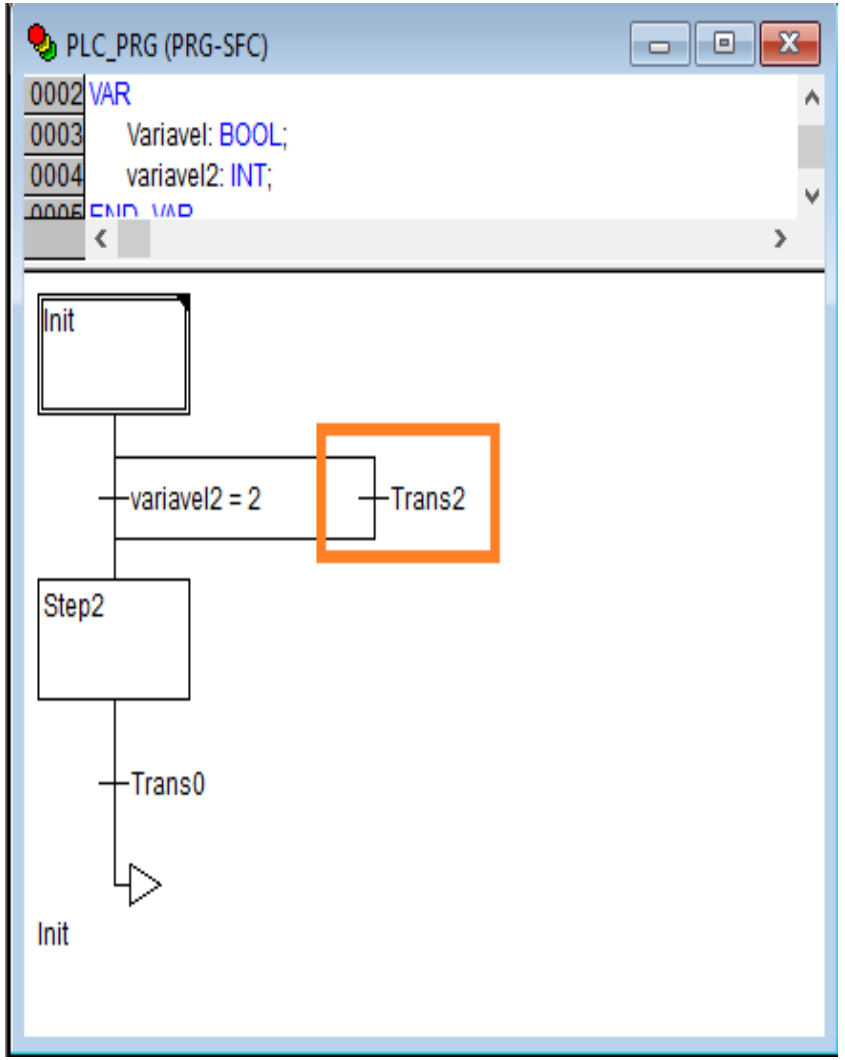
Poderíamos ter mais blocos em nosso diagrama

Ramificação



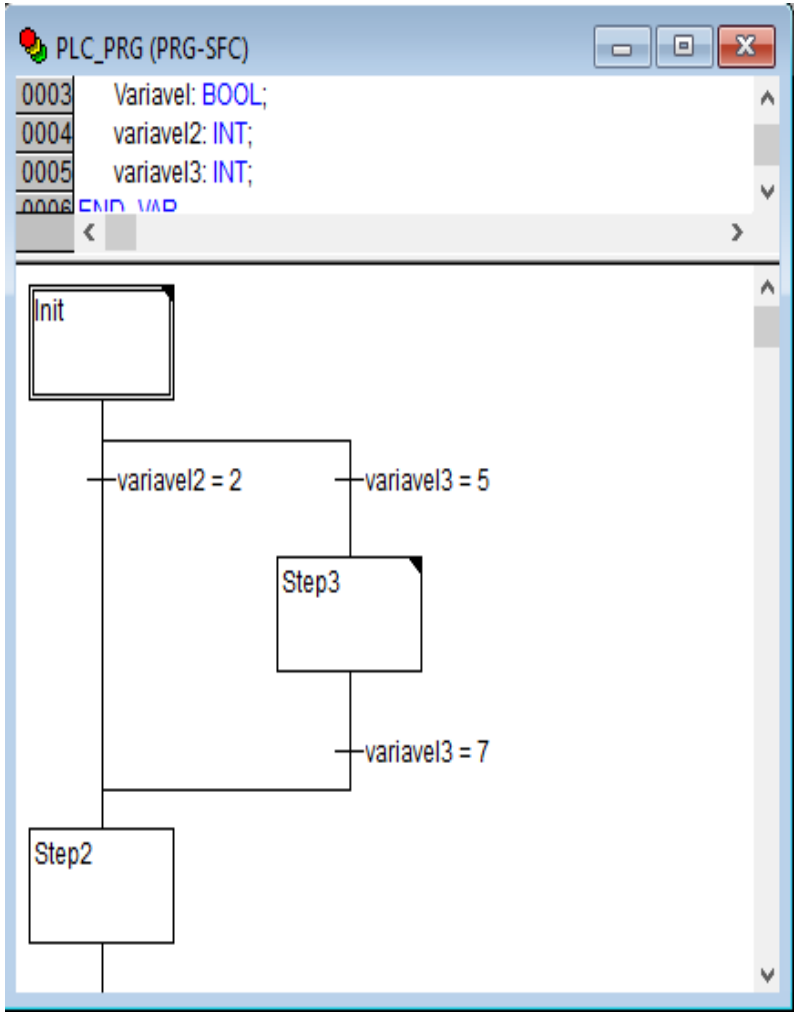
Ao selecionarmos um objeto do diagrama e clicarmos no ícone azul adicionamos um ramo concorrente no nosso diagrama.

Em laranja podemos observar o novo ramo composto por uma condição concorrente á primeira.



Podemos adicionar vários blocos á esse novo ramo

Exemplo de ramificação com Blocos



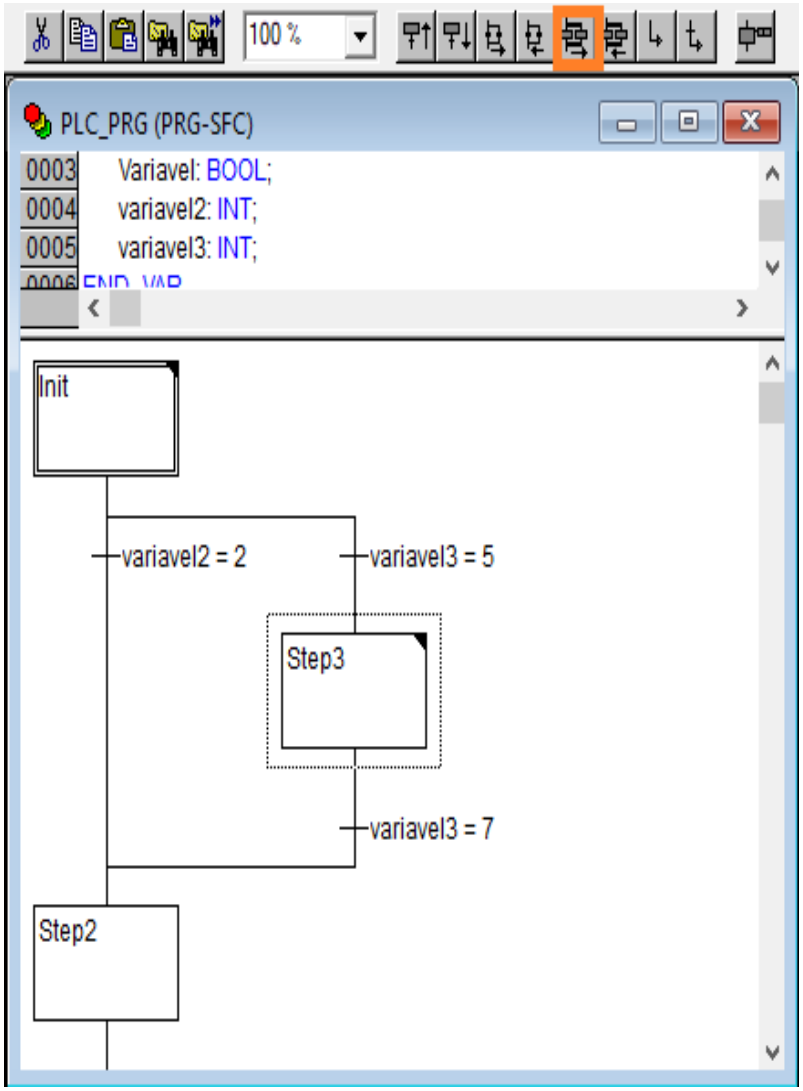
Podemos então adicionar blocos ás nossas ramificações, dessa forma o programa passa a ter vários caminhos disponíveis de processamento.

Nesse caso, o programa rodará o init. Se a condição variavel2=2 for atendida, o programa prosseguirá para o bloco Step2, se a condição variavel3 = 5 for atendida, o programa executará primeiramente o Step3 até que a condição variavel3 = 7 for atendida.

Se mais de uma condição for atendida, o programa dará prioridade da esquerda para a direita.

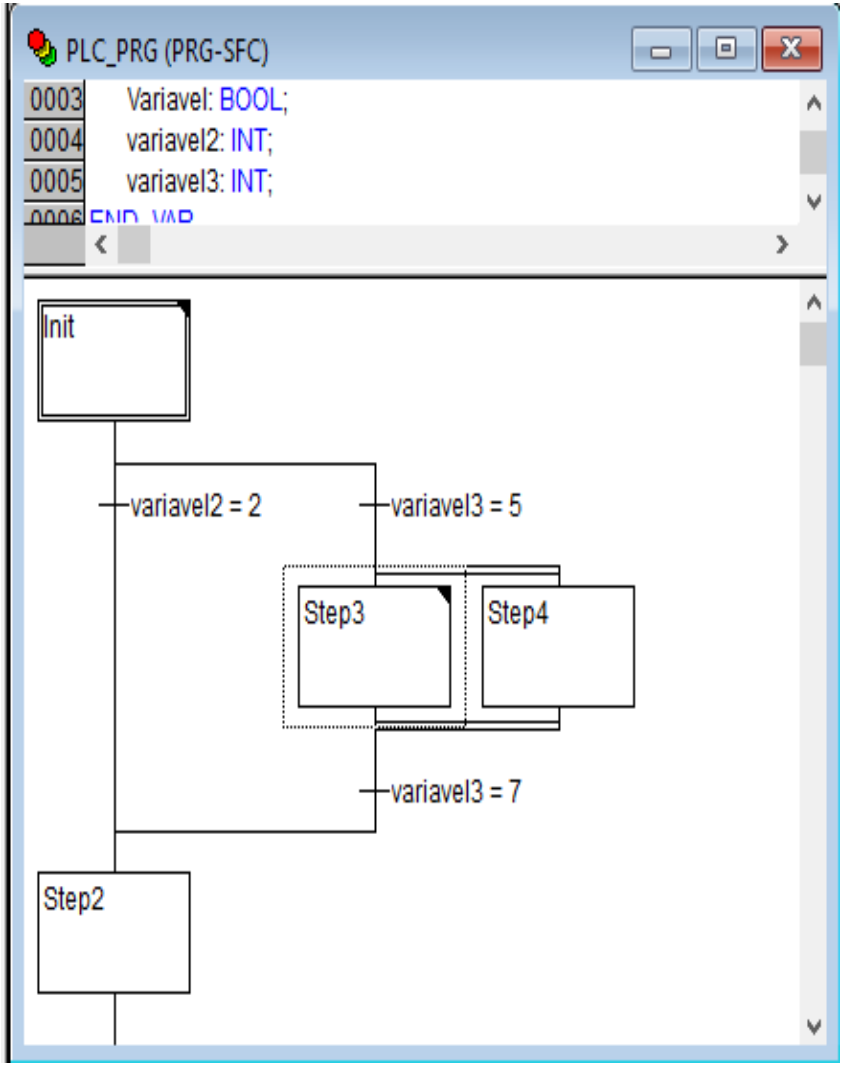
Lembrando que o programa ficará preso á ramificação até que termine a mesma

Paralelismo

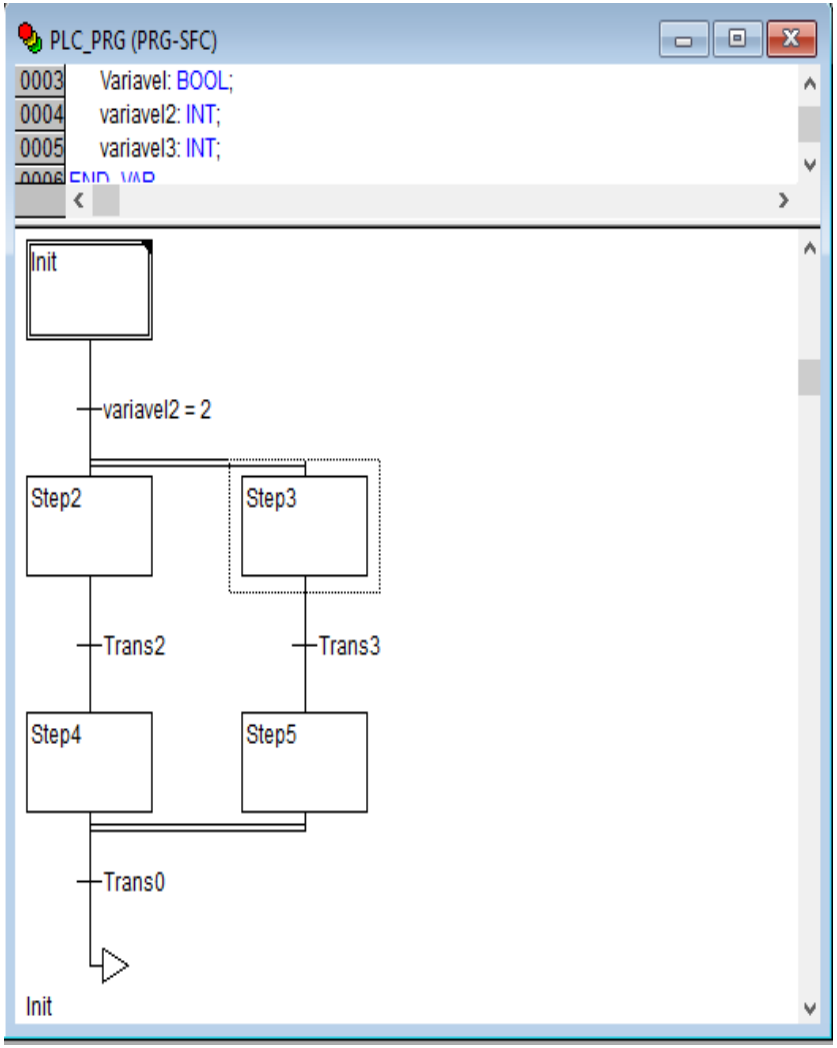


Ao selecionarmos um bloco e clicarmos no ícone em laranja adicionamos um bloco em paralelo com o primeiro.

O novo bloco Step4 está em paralelo ao Step3. Eles serão executados simultaneamente



Sincronismo

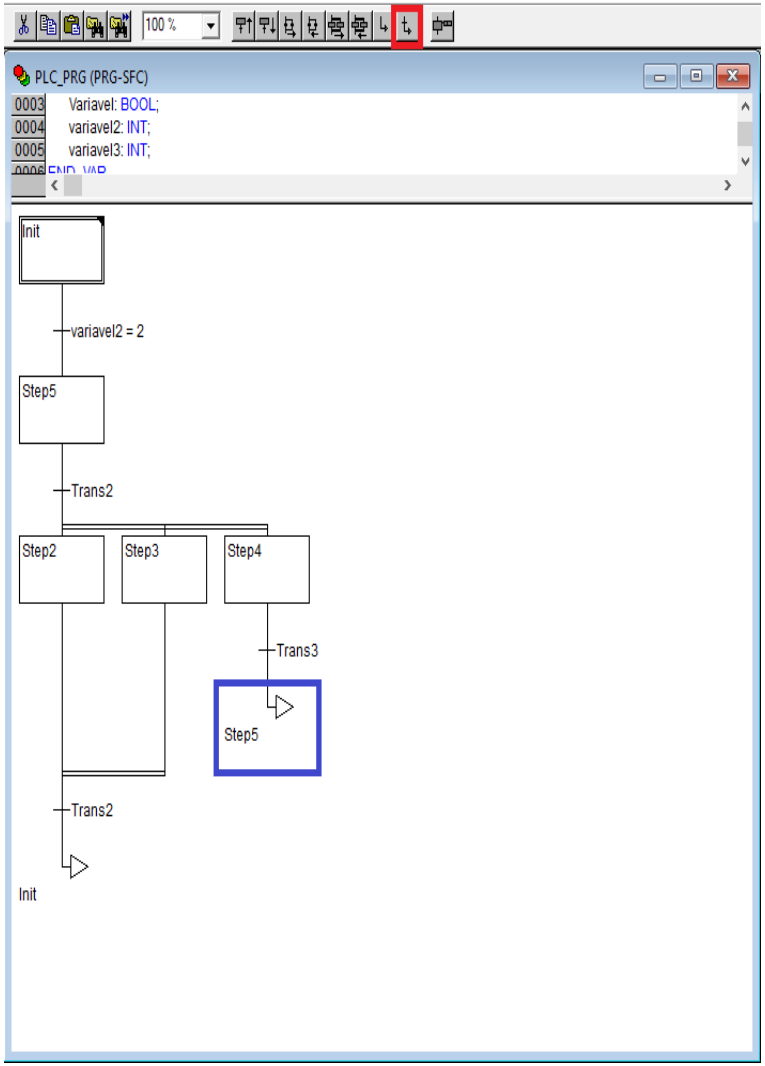


Em processos paralelos há um sincronismo. Desta forma o programa só voltará ao ramo principal após a todos os processos do ramo paralelo serem realizados.

Nesse exemplo, a condição Trans0 só passará a funcionar se o processo do ramo esquerdo estiver em Step4 e o processo do ramo direito estiver em Step5.

Se um dos processos não estiverem no final, o programa ficará preso até que esse o chegue.

Salto



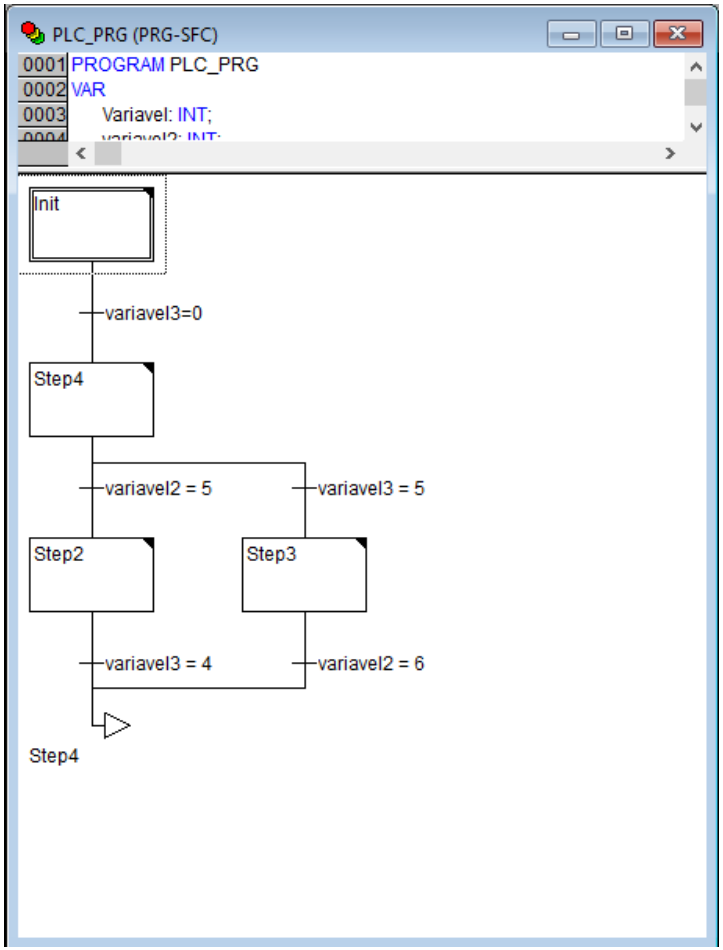
Podemos também fazer saltos entre processos em nosso programa

Ao selecionarmos um bloco e pressionarmos o ícone em azul adicionamos um salto que leva até outro bloco.

Nesse exemplo, ao executarmos o Step4 e a condição Trans3 for atendida o programa retornará ao Step5.

O icone a esquerda do marcado em vermelho é um salto sem condição

Exemplo final



Bloco init:
Variavel2 := 10;
Variavel3 := 0;

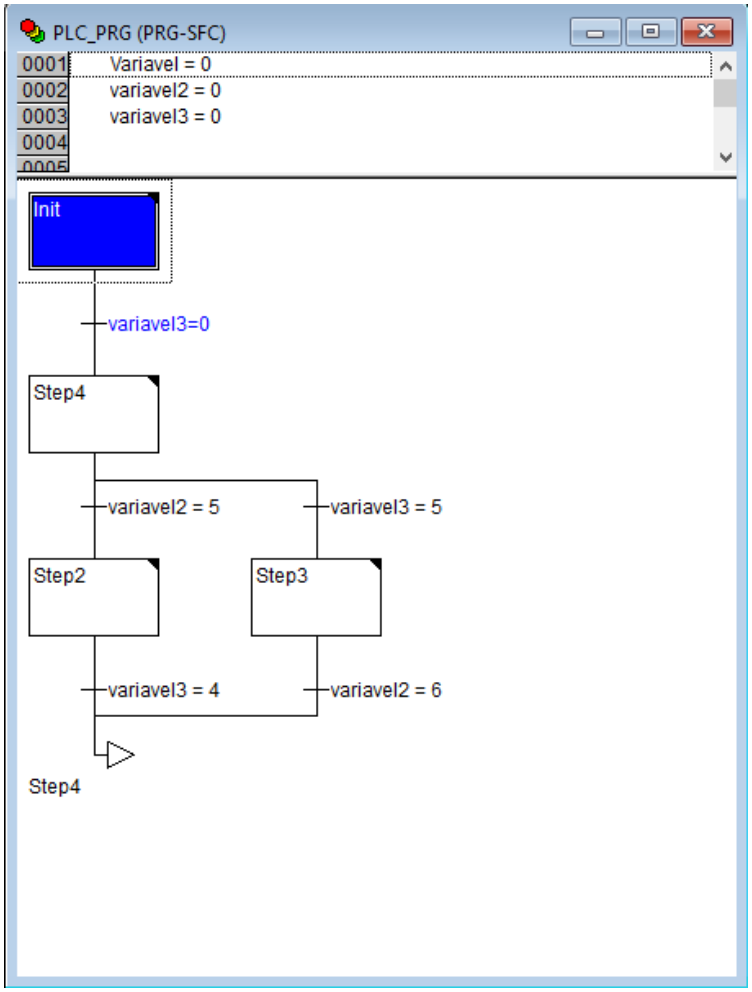
Bloco Step4:
Variavel2 := variavel2 – 1;
Variavel3 := variavel3 + 1;

Bloco Step2:
Variavel3 := 4;

Bloco Step3:
Variavel2 := 6;

Exemplo

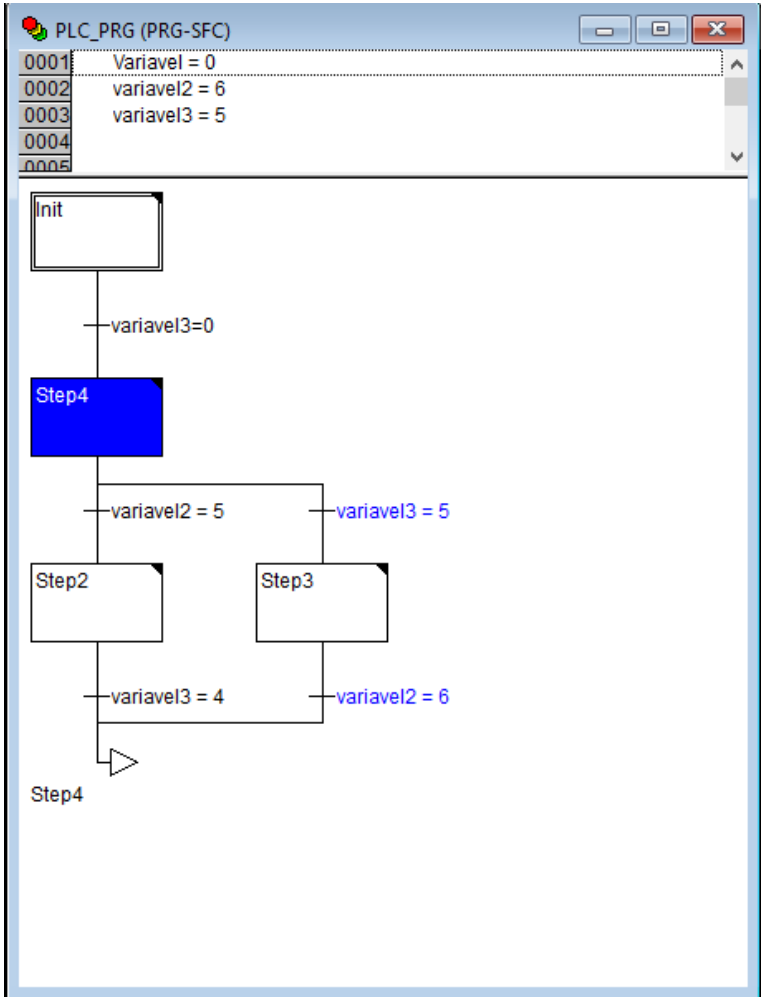
Clock 1



Apesar de em init o valor de variavel2 ser setado para 10, como a condição variavel3 = 0 já está sendo satisfeita, o programa ignora o que está em init

Assim variave3 chega em 5 primeiro e é executada primeiro.

Clock 2



Exemplo

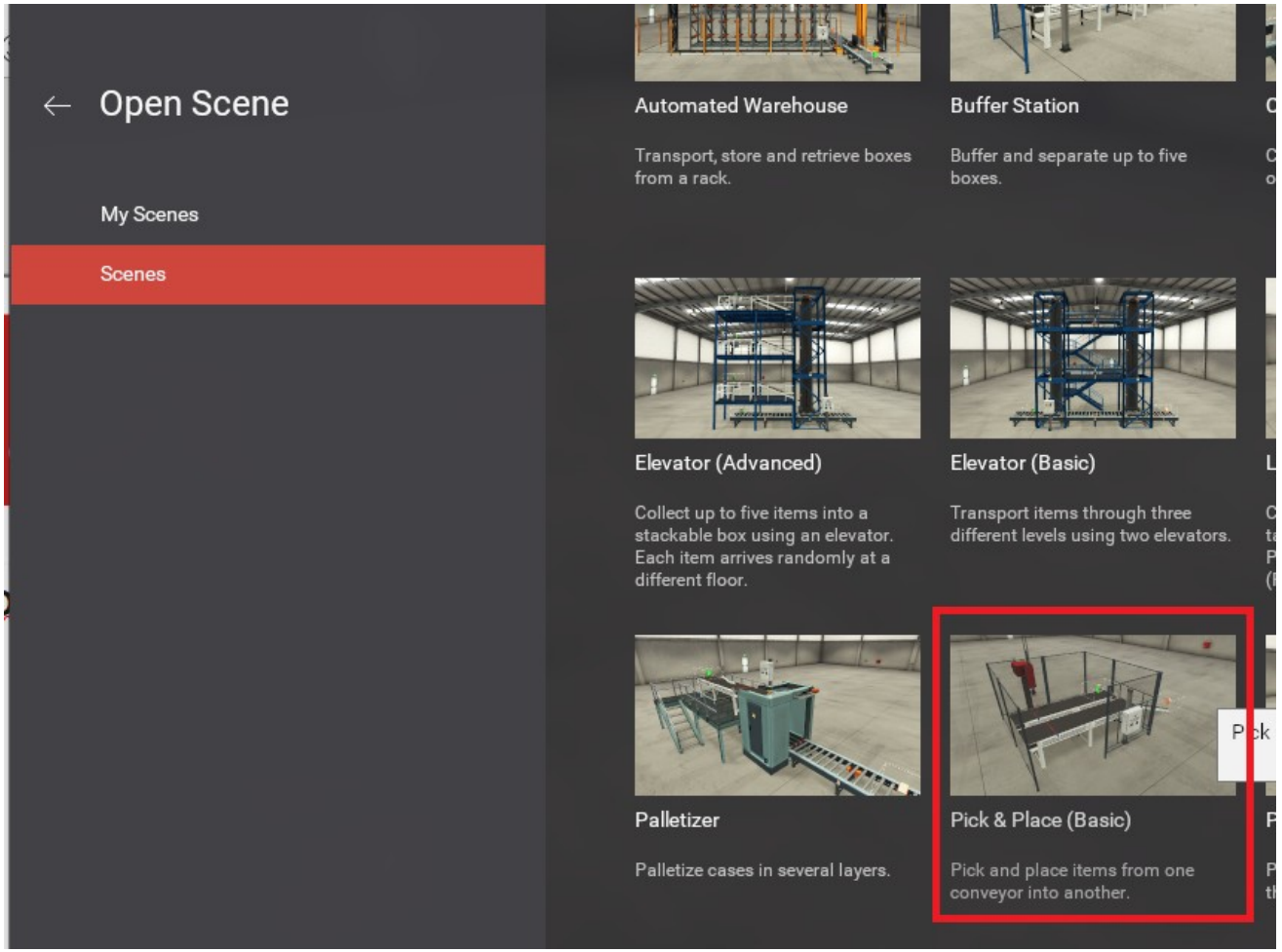
Após isso o programa ficara infinitamente alternando entre Step2 e Step3.

Aplicação no Factory IO

Agora iremos aplicar a programação SFC na cena Pick & Place (basic) do factory IO. Para isso, devemos criar um novo programa SFC no CodeSys, abrir a cena no Factory IO e fazer as comunicações OPC e PLC.

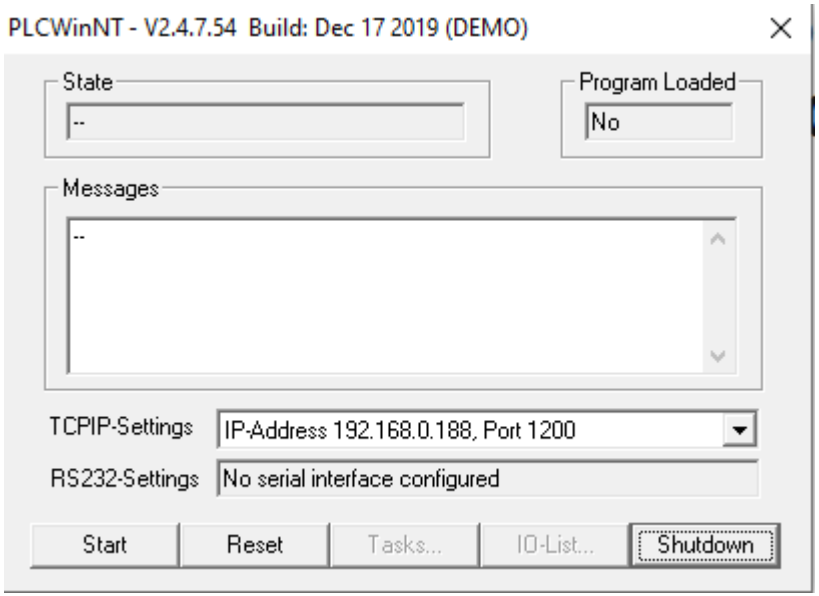
Aplicação no Factory IO

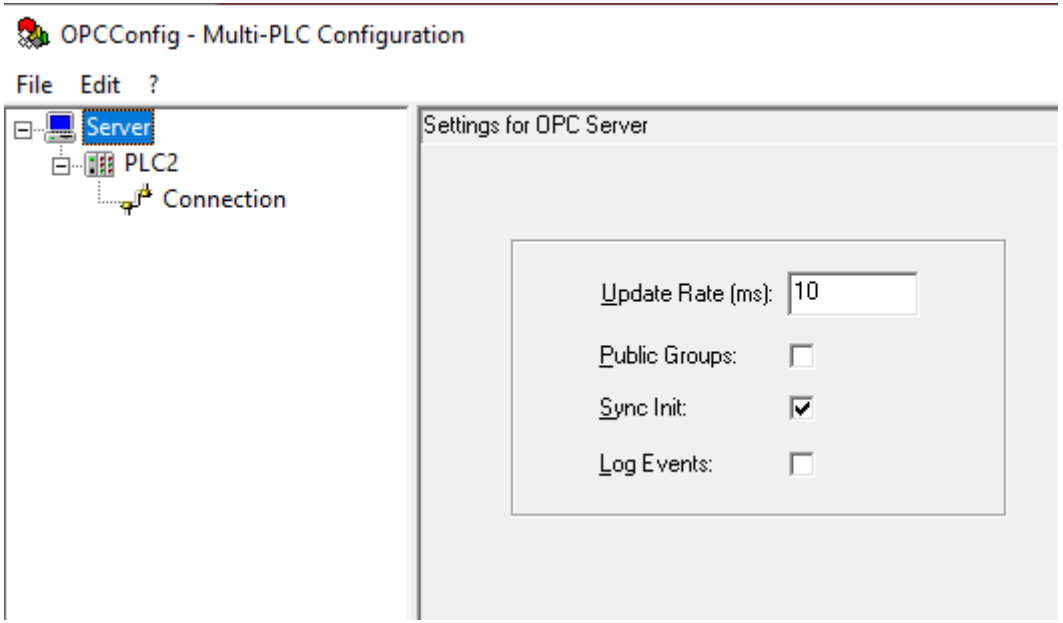
No factory IO clicamos em scenes e depois em Pick & Place (basic):



Conexão

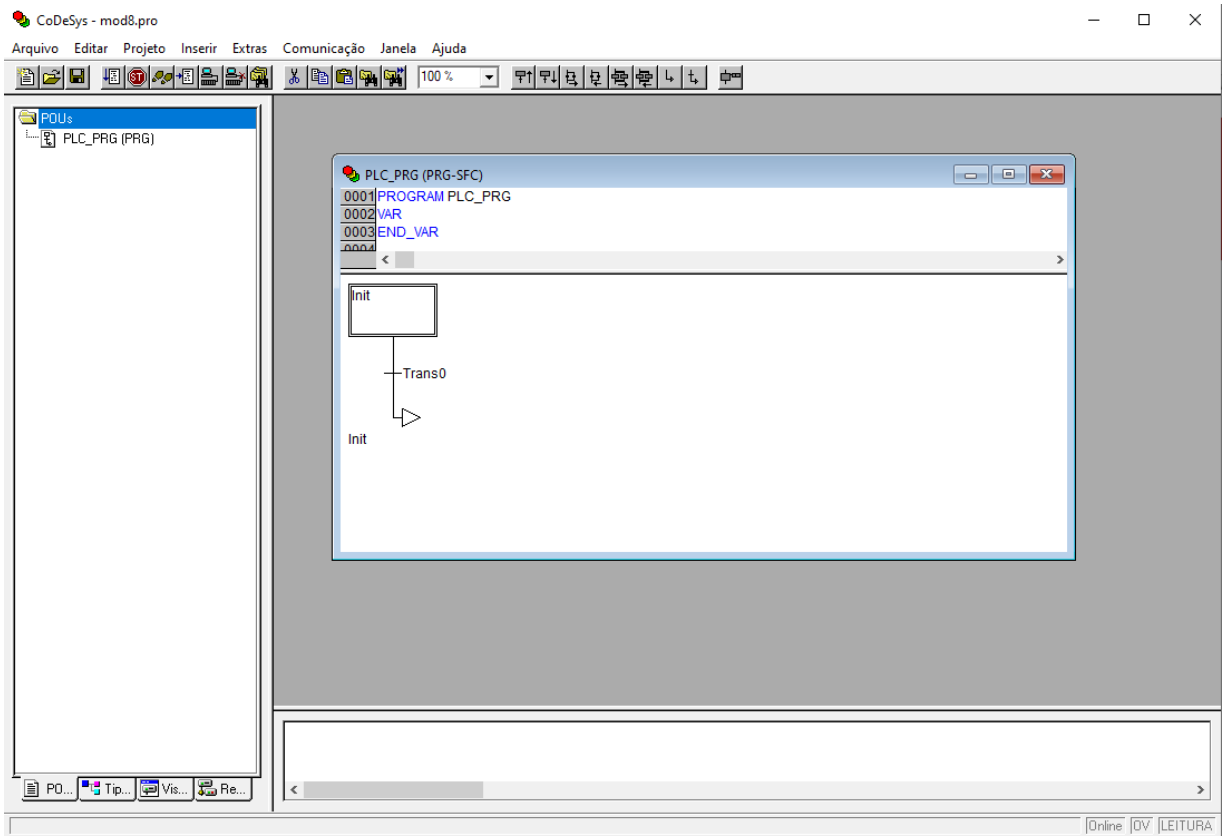
Abrimos o OPC Konfigurator e o PLC WIN NT e fazemos as conexões iniciais (lembrando que a porta deve ser 1200):





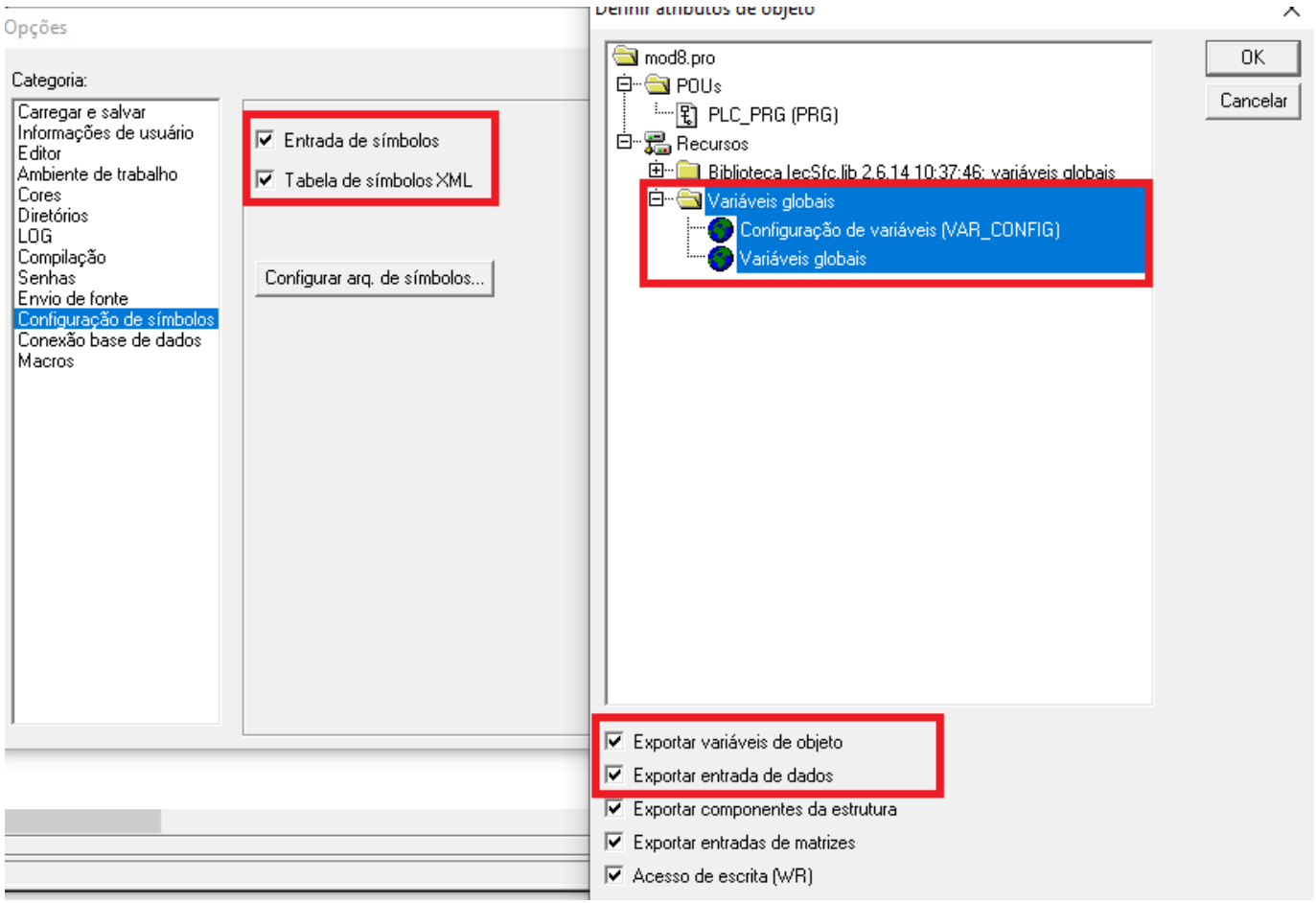
Criando novo projeto no CodeSys

Vamos criar um novo projeto no CodeSys da mesma maneira apresentada no começo desse tutorial:



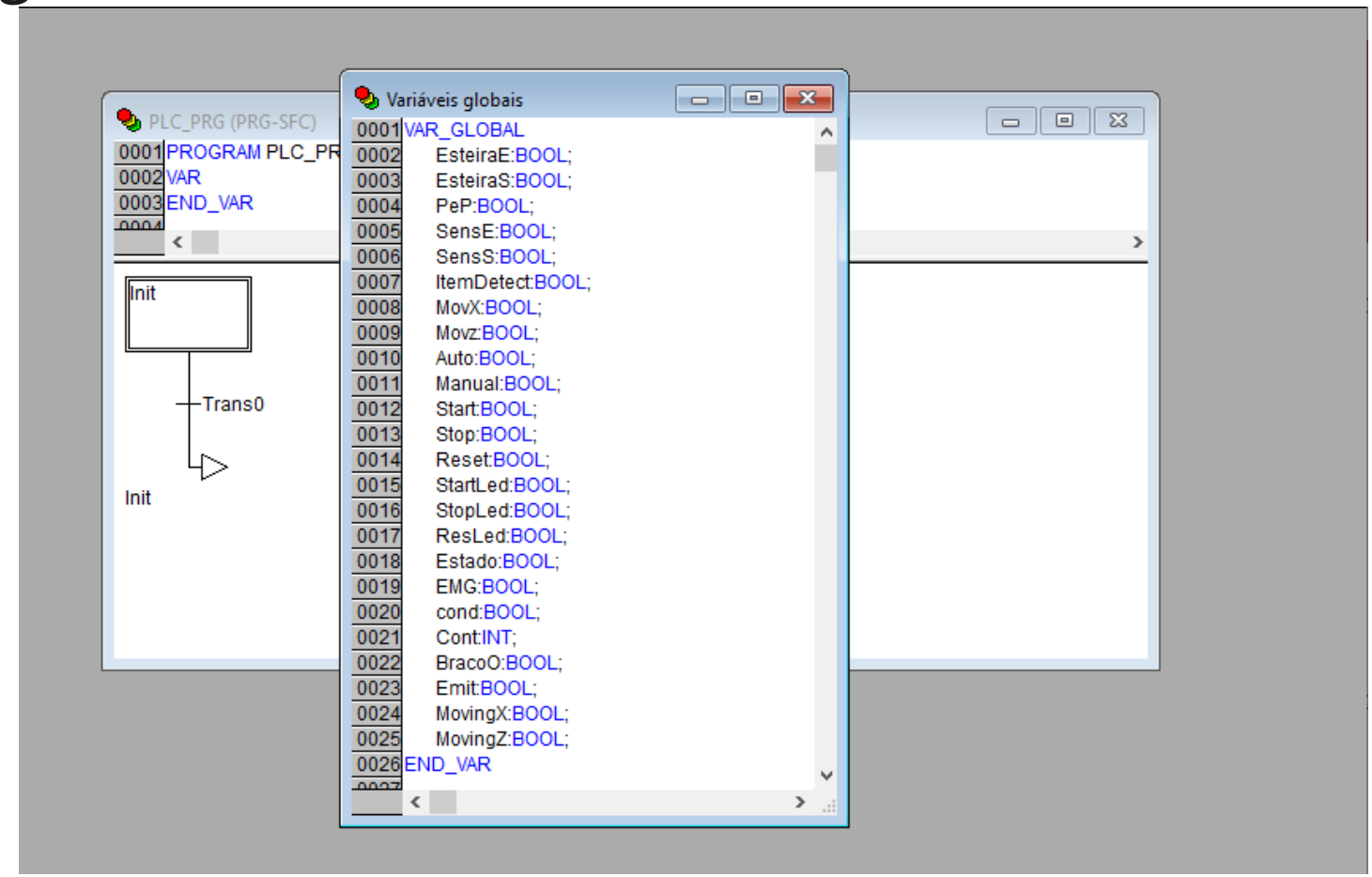
Criando novo projeto no CodeSys

Antes de iniciarmos a programação, vamos exportar as variáveis globais:



Programando

Assim como o programa em Ladder, vamos utilizar as mesmas variáveis globais:

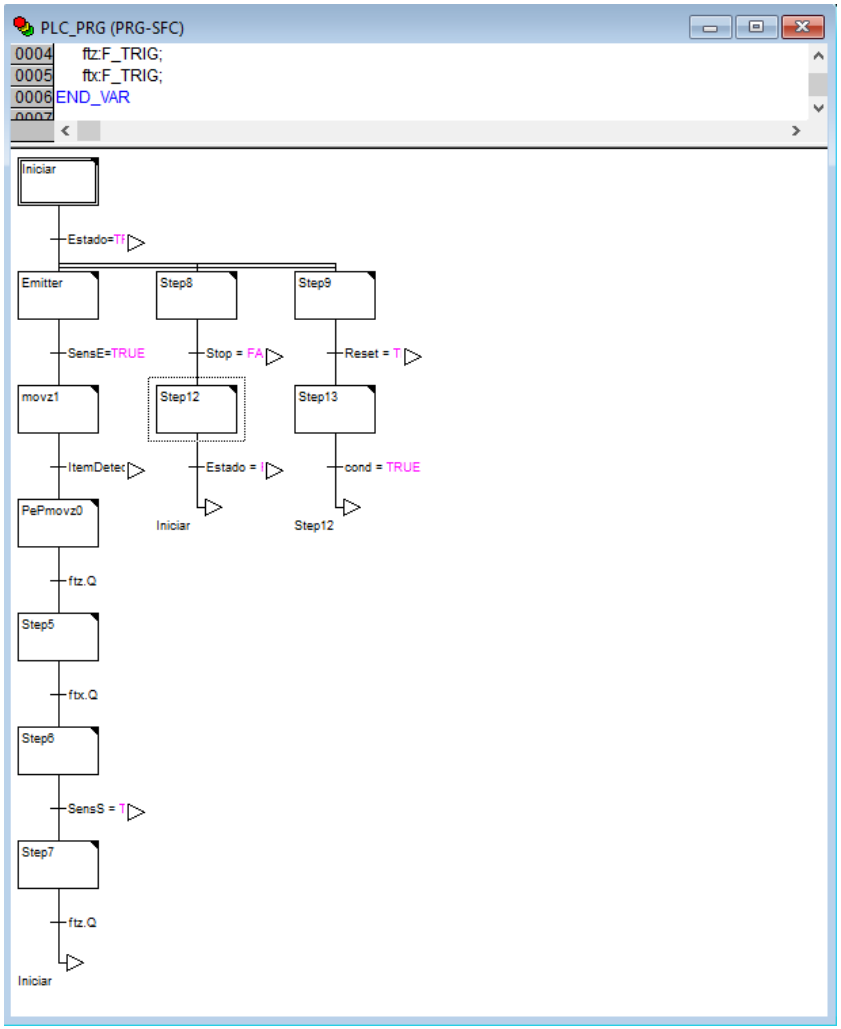


Programando

No Factory IO ligamos as variáveis do CodeSys às tags do factory IO:

Do programa

Para controlar essa cena, podemos usar a seguinte programação em SFC:

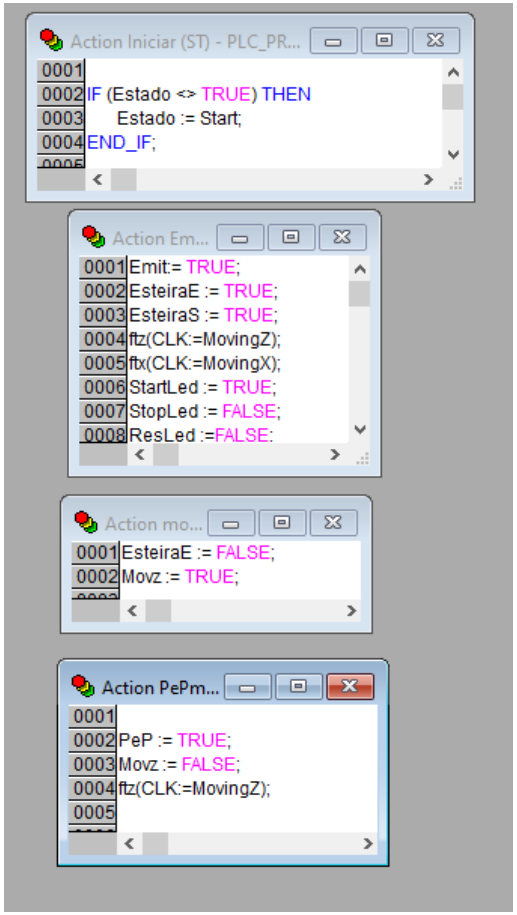


Essa programação consiste em 3 ramificações paralelas. A primeira consiste no processo ativo quando apertamos o botão de Start, a segunda consiste na paralização quando apertamos o botão Stop e a terceira no reset.

Além disso, declaramos duas variáveis (ftz e ftx) do tipo Fallen Trigger (F_TRIG) para podermos captar a descida dos sensores de movimento no eixo x e y da garra.

Do programa

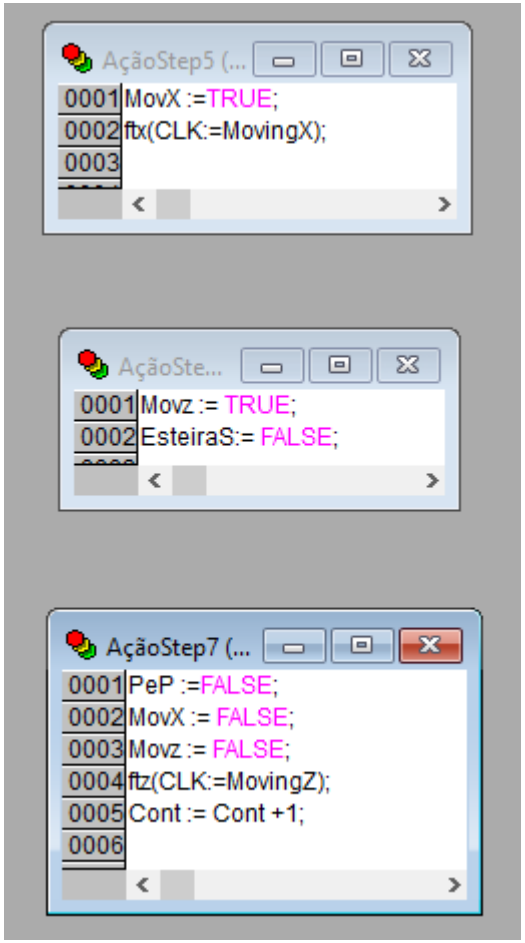
Vamos analisar o primeiro ramo:



Em ordem podemos ver os códigos de cada bloco.

- O primeiro bloco (iniciar) espera até que o botão de Start seja apertado trocando o valor de estado de 0 (desligado) para 1 (ligado);
- Quando o estado está em 1, o segundo bloco liga o emitter, as esteiras e o Led do botão Start, além de ligar as variáveis do tipo F_TRIG com os sensores de movimento da garra;
- Quando o Sensor de entrada detecta um item, entramos no bloco 3, o qual para o a esteira de entrada e desce a garra;
- Quando o item é detectado, a garra é ativada e capta o item. A garra se move para cima e atrela a variável ftz ao sensor de movimento da garra no eixo .

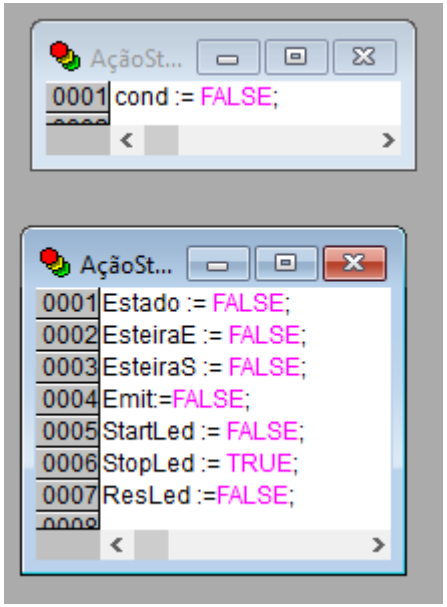
Do programa



- Com ftz.Q, a condição de transição entre o bloco 4 e 5 é atingida. Desta forma, quando a garra termina de deslocar para cima, MovingZ que estava em 1 vai para 0 sendo captado a borda de descida na função Q.
- No bloco 5, A garra se desloca no eixo X em direção à esteira de saída;
- Mais uma vez usamos a função Q para identificar o término do deslocamento do movimento da garra e após isso a mesma começa a se mover para baixo e a esteira de saída desliga;
- Ao ser detectado o item no sensor de saída, a garra solta o item e retorna à posição inicial. O contador aumenta a contagem em 1.
- Ao retornar à posição inicial, o programa retorna ao inicio.

Do programa

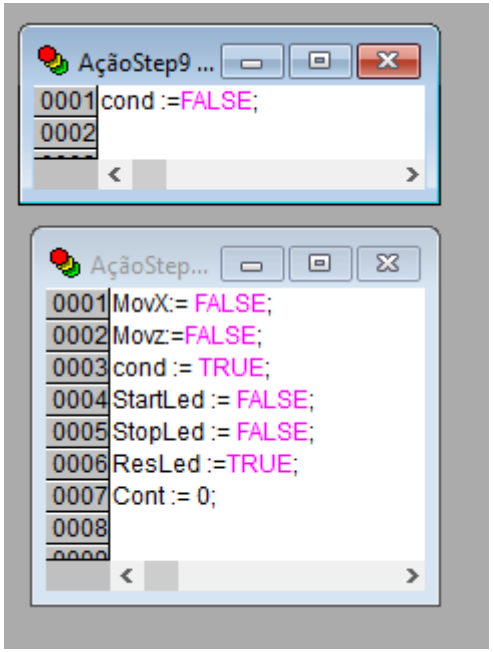
Agora vamos analisar o segunda ramo:



O programa espera o botão Stop ser pressionado. Quando o mesmo é pressionado, o programa desliga todos os componentes e volta ao inicio do programa.

Do programa

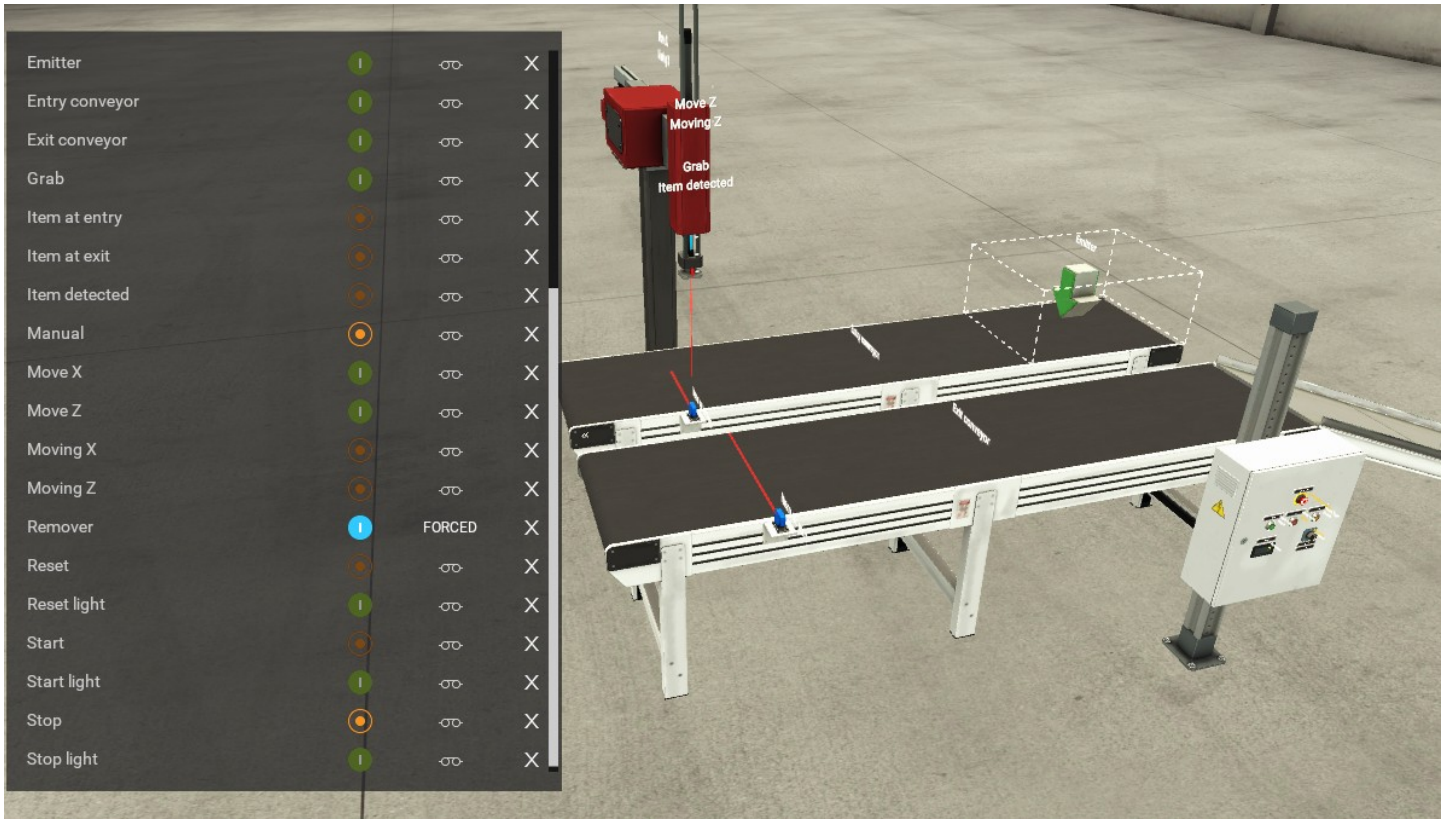
Agora vamos analisar o terceiro ramo:



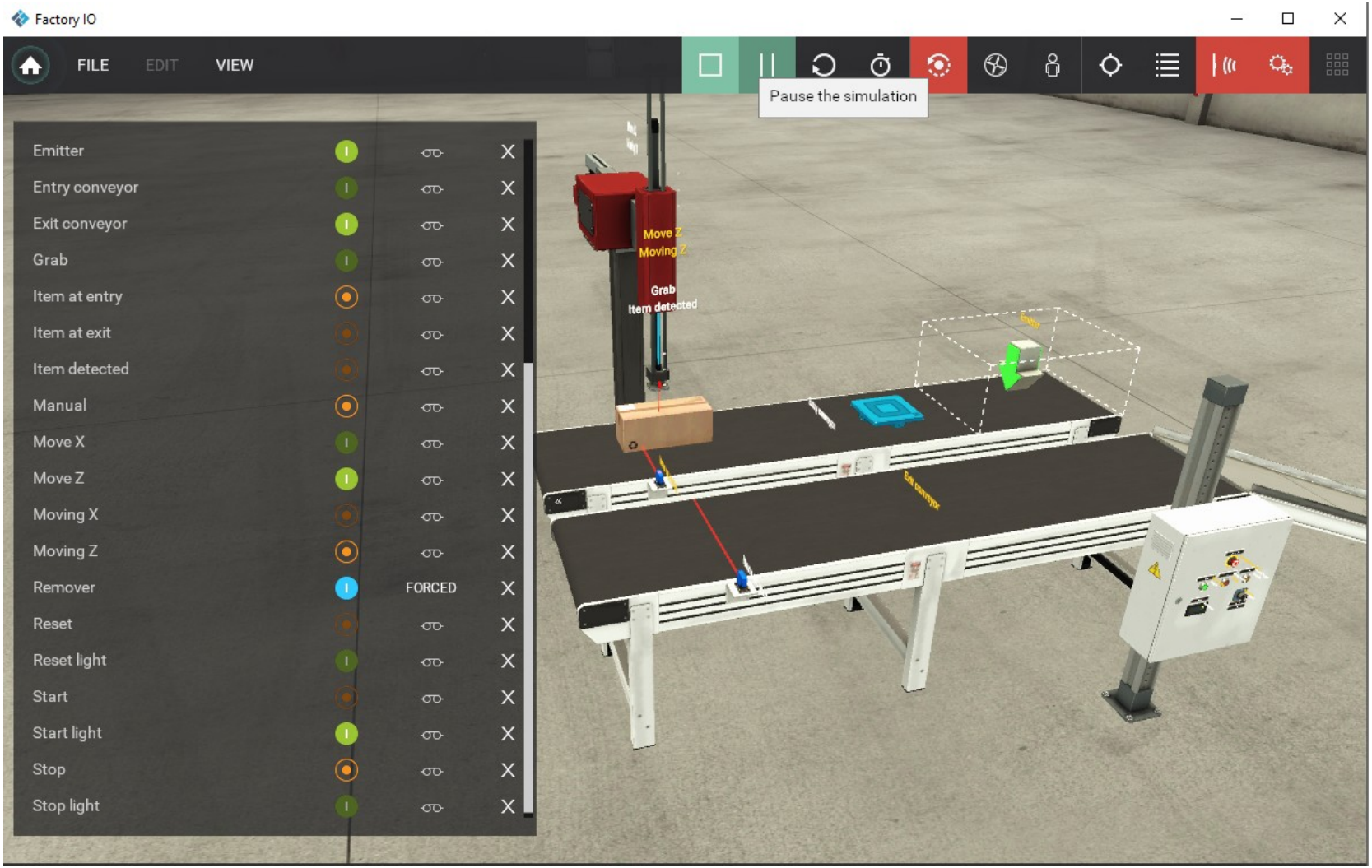
O programa espera o botão reset ser pressionado. Ao ser pressionado, o programa para e retorna os componentes às condições iniciais.
As esteiras param, a garra volta à posição inicial e o contador é resetado.

No Factory IO

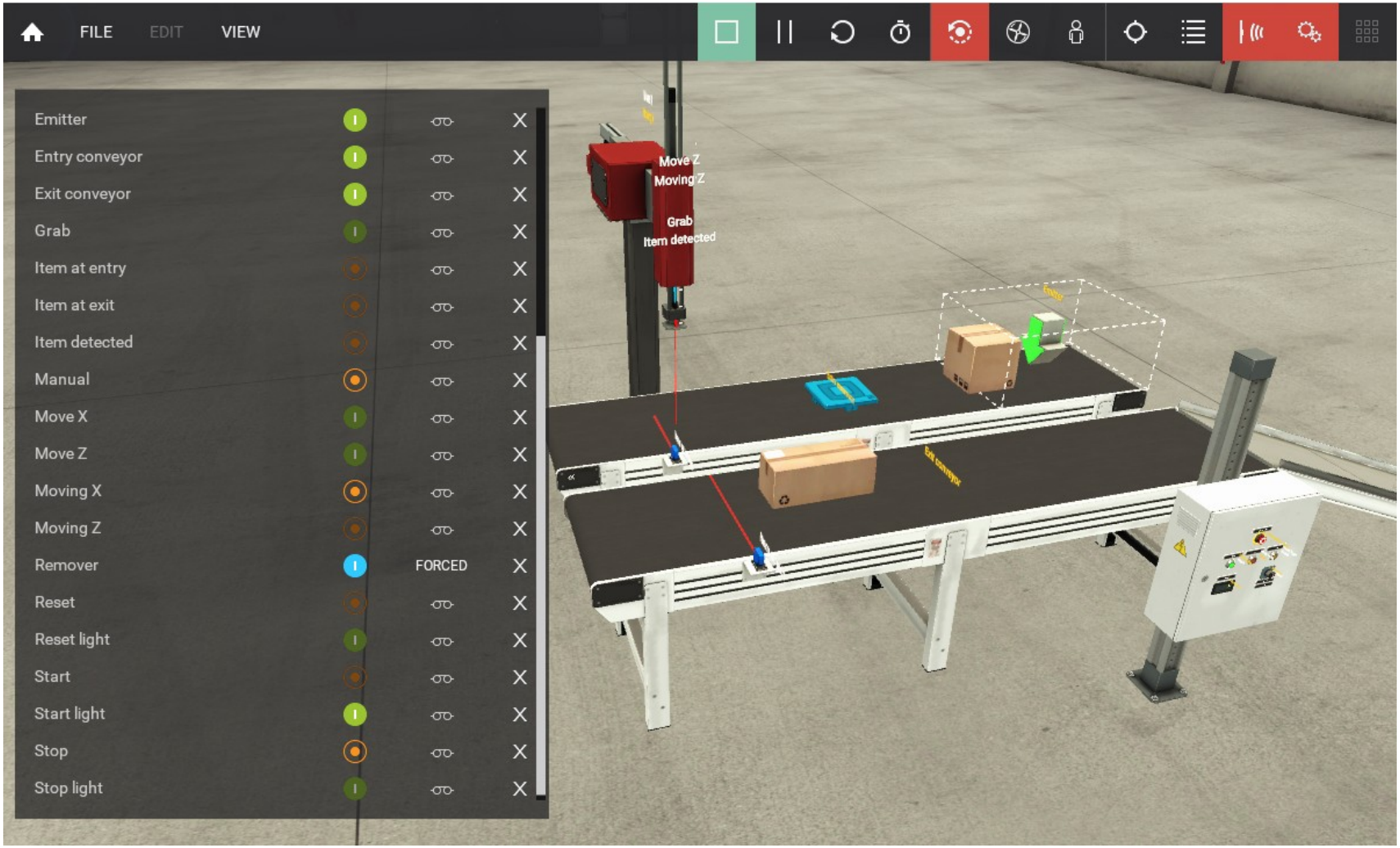
Podemos ver agora no Factory IO que a cena funciona perfeitamente



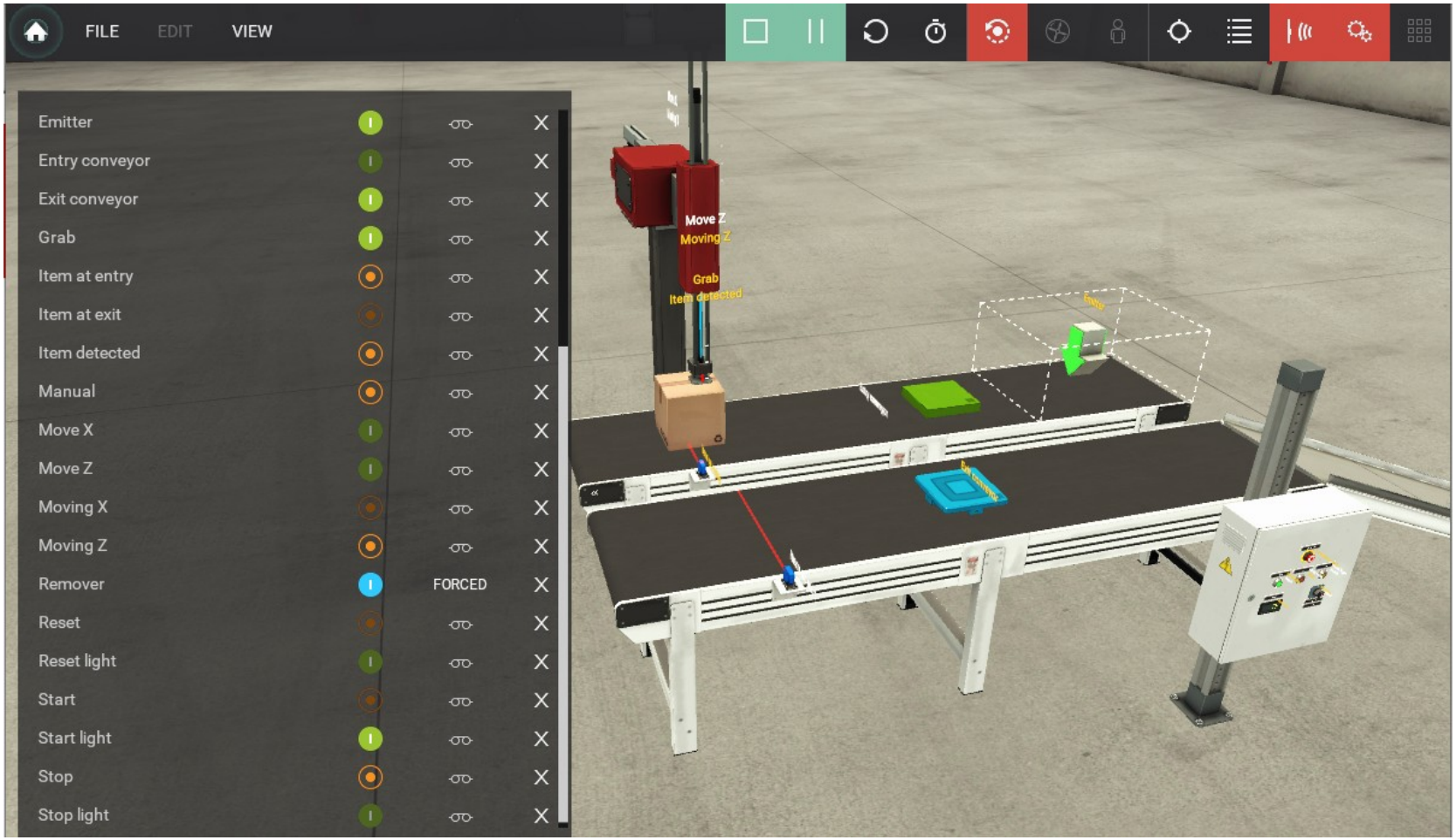
No Factory IO



No Factory IO

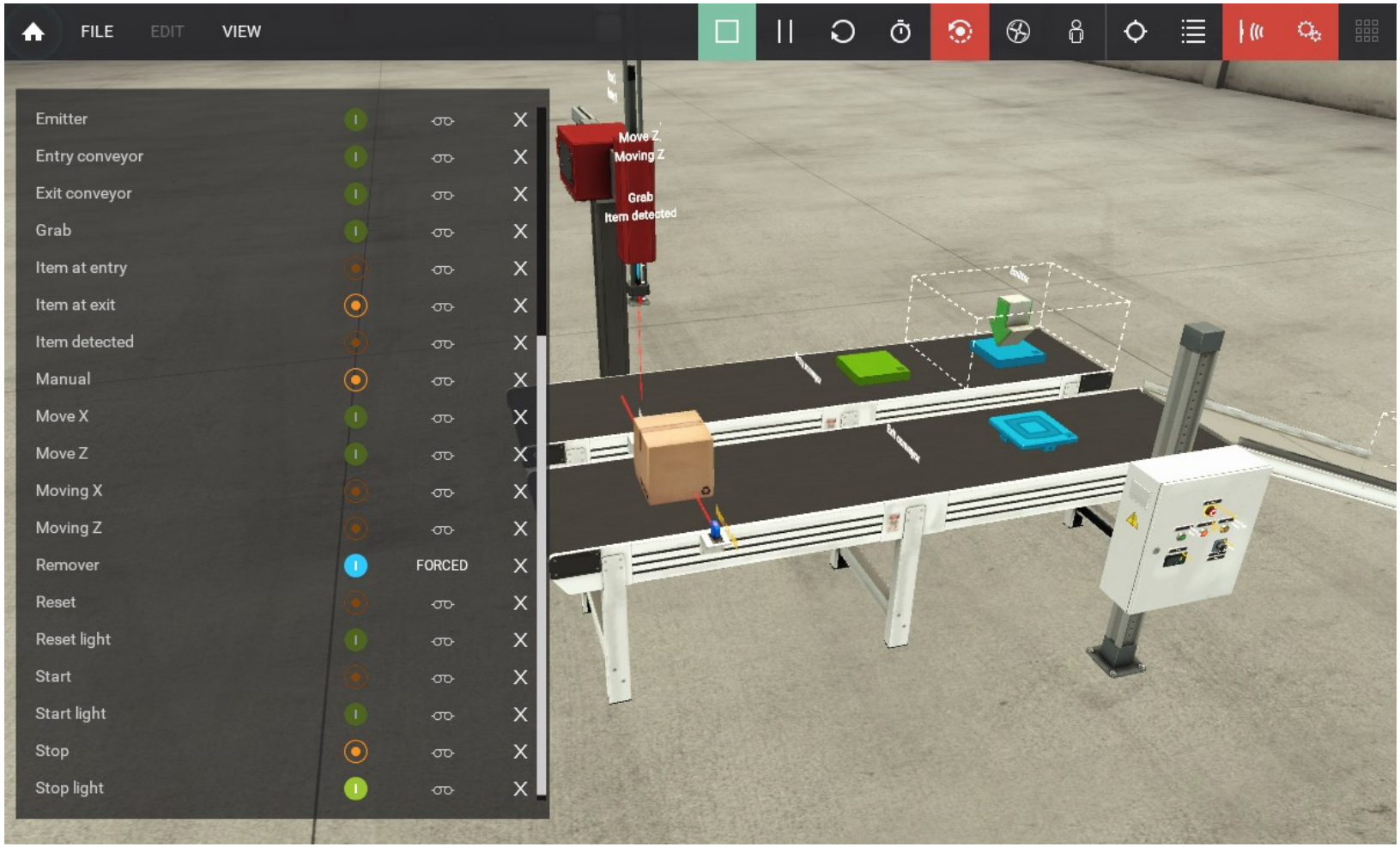


No Factory IO



No Factory IO

Ao pressionar o botão de stop o processo para:



No Factory IO

Fim