SUPORTE DE CURSO

IEC 1131-3 Sequential Charts

Livro Texto: Programming industrial control systems using IEC 1131-3 – R.W. Lewis

UFMG – Informática Industrial
Prof. Constantino Seixas Filho

IEC 1131-3 Sequential Function Charts

Structured Text (ST)	Textuais
Instruction List (IL)	
Function Block Diagram (FBD)	Gráficas
Ladder Diagram (LD)	
Sequential Function Charts (SFC)	

SFC				
ST	IL	LD	FBD	
TEXTUAIS		GRÁFICAS		

É usada para:

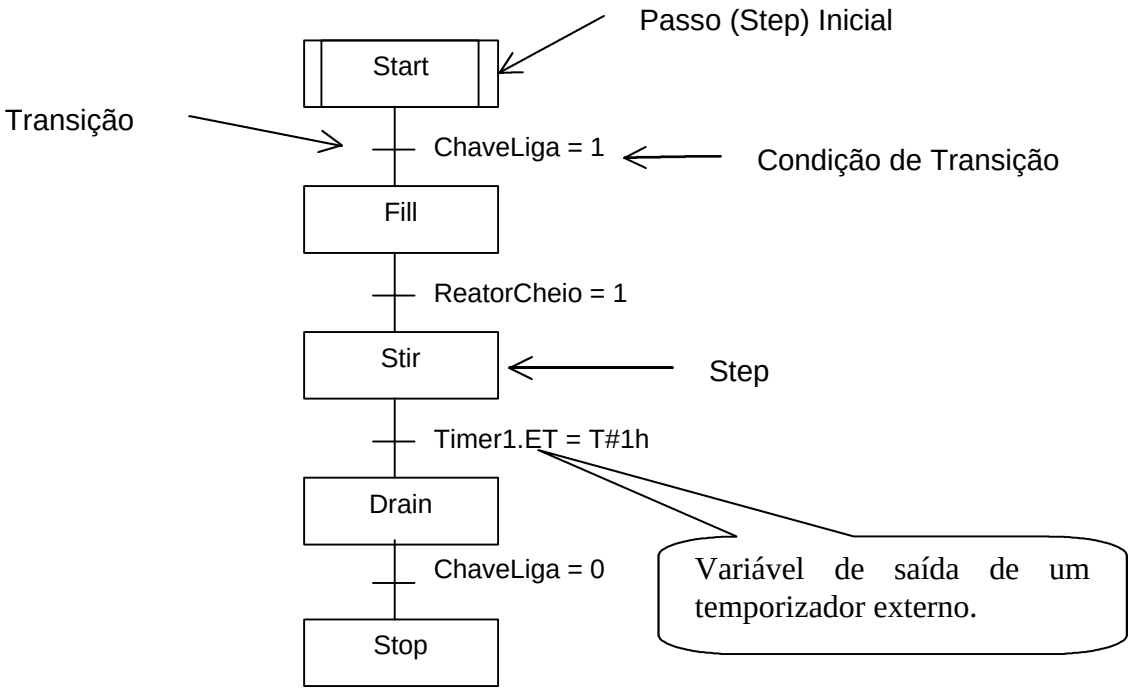
- Descrever o comportamento seqüencial de um sistema.
- Como linguagem de estruturação das ações de um programa segundo um modelo *top-down*. Este particionamento do problema traz ganhos de performance porque apenas o código relativo aos passos ativos é executado.
- Para descrever o comportamento baixo nível de um processo seqüencial.
- Para representar as fases de um processo de batelada.
- Para representar um processo de comunicação de dados, etc.

Histórico

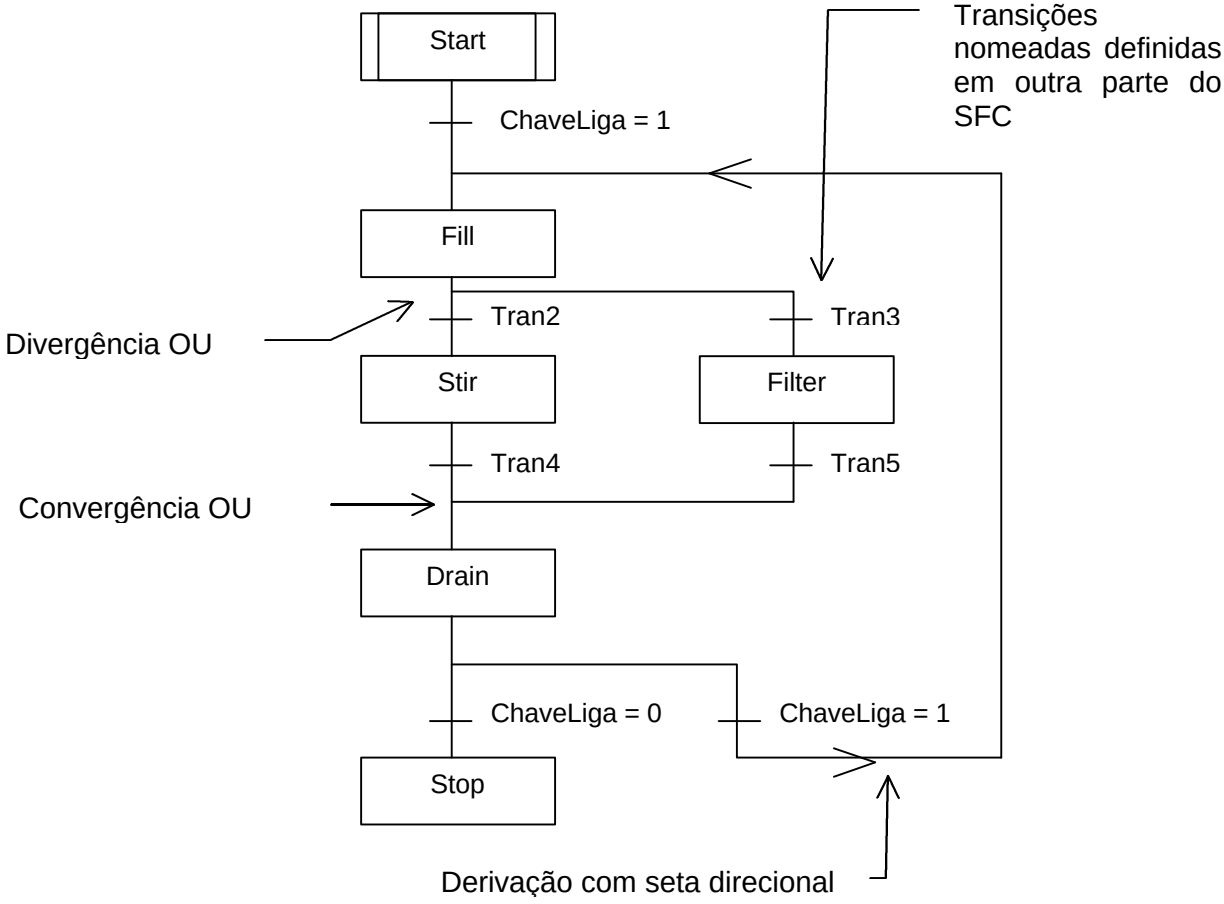
Universidades francesas desenvolveram uma linguagem de representação de processos seqüenciais baseada nas Redes de Petri: o Grafcet (*Graphe Fonctionnel de Command Etape Trasition*). Grafcet se tornou um padrão europeu. Em 1988 foi publicado o padrão IEC 848: *Preparation of function charts for control system*, baseado na linguagem Grafcet.

A norma IEC 61131-3 introduziu pequenas modificações no padrão IEC 848 visando acoplar esta quinta linguagem às demais linguagens da suite 1131-3.

Estrutura de um chart

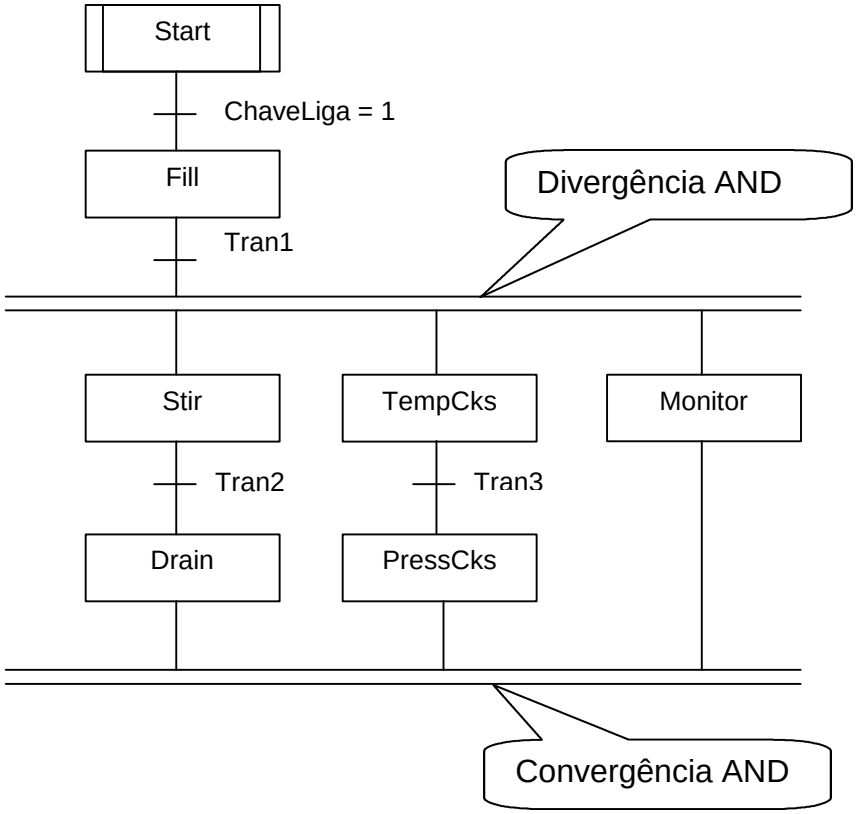


Caminhos convergentes e divergentes



Seqüências simultâneas

Faz uso da construção divergência AND para iniciar ações que serão realizadas em paralelo. e convergência AND que espera até todas as tarefas estarem concluídas.



Steps

Cada Step deve receber um nome único e só pode aparecer uma única vez em uma SFC. Os nomes dos Steps e transições são locais a uma POU (bloco de função ou programa) onde a SFC é definida.

Existem dois tipos de Steps:

1. Steps normais: são exibidos em caixas retangulares com o nome do step no centro.
2. Step inicial: Possui barras verticais nas laterais do retângulo. Todo SFC deve ter um step inicial. O step inicial deve ser único.

Cada Step pode ter um bloco de ação associado. O bloco de ação pode ser definido utilizando-se qualquer uma das outras linguagens IEC: ST, FBD, LD ou IL.

Cada estado possui duas variáveis associadas:

Flag de estado ativo:

É uma variável booleana que só está ativa quando um step em particular está ativo.

Sintaxe: <StepName>.X

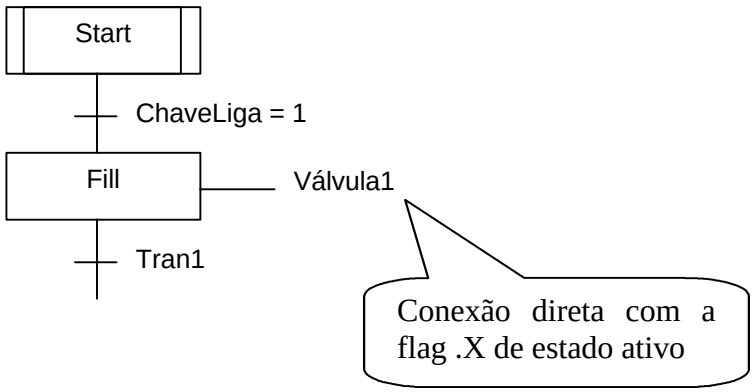
Exemplo

Na figura anterior podemos definir Tran3 em linguagem ST:

Drain.X = 1

A transição do step ***TempCks*** para o step ***PressCks*** só ocorrerá quando o estado ***Drain*** for ativado.

Esta flag pode ser conectada diretamente a uma saída booleana:



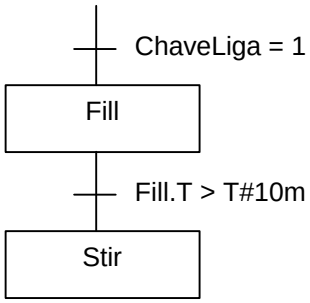
Flag de tempo decorrido

Cada estado tem associada uma variável de tempo decorrido do tipo TIME. Esta variável computa quanto tempo o estado está ativo. Ela é zerada toda vez que o estado é ativado. Esta variável retém o seu último valor quando o estado é desativado.

Sintaxe: <StepName>.T

Esta variável é útil para se determinar por quanto tempo um determinado estado está ativo.

E x e m p l o :



O SFC ficará no estado Fill durante 10 minutos.

Uma outra função desta variável pode ser a de diagnóstico, pois podemos avaliar por quanto tempo cada estado de uma seqüência ficou ativo.

Transições

Cada transição deve ter uma condição associada. Se a transição deve ocorrer sempre, então devemos associar a ela a condição TRUE.

Exemplos Diversas maneiras de se representar uma transição

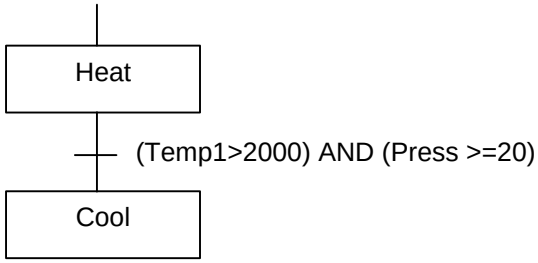
Texto Estruturado	
	Qualquer expressão ST que resulte em um valor booleano pode ser usada para descrever uma condição de transição.

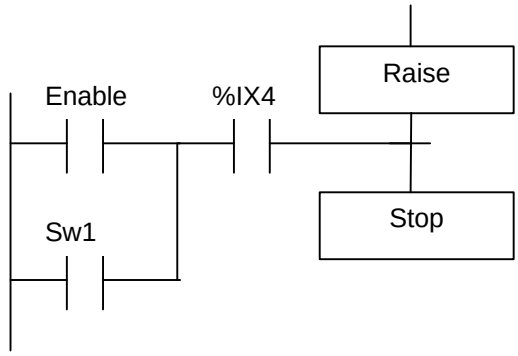
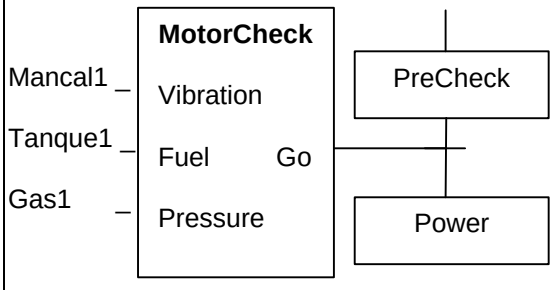
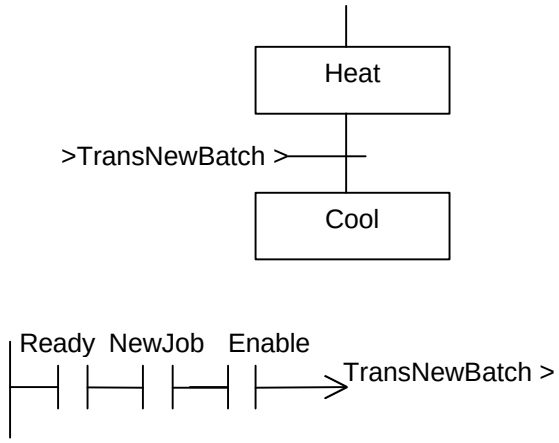
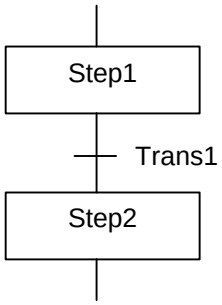
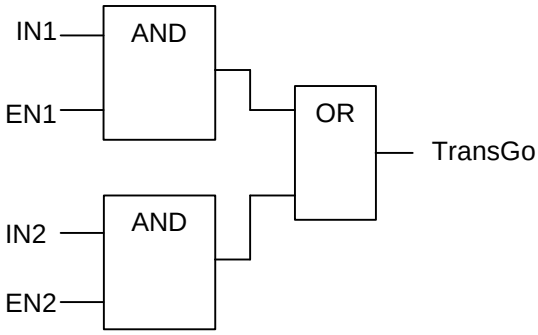
Diagrama Ladder	
	Quando um degrau permite a passagem de potência, a condição de transição se torna verdadeira.

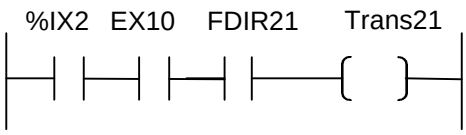
Diagrama de Blocos de Função	
	Qualquer FBD que produza um resultado booleano, pode ser ligado a uma transição. No exemplo ao lado o bloco MotroCheck irá fazer GO =1 se os valores de vibração, combustível e pressão estiverem dentro de seus limites operacionais.

Conector de transição	
	Usamos um conector de transição para colocar a condição de transição em outra parte do diagrama.

Transição nomeada	
	Uma transição pode receber um nome e ser definida em outra parte do diagrama usando uma das linguagens IEC: ST, FBD, LD ou IL

Transição definida usando ST	
TRANSITION Trans1: := AB1 AND CX3 OR CX5 AND (TX3 >= 100.2); END_TRANSITION	A expressão deve retornar um valor booleano. Note que o símbolo de atribuição := é necessário para indicar que o valor é atribuído para a condição de transição.

Transição definida usando FBD	
TRANSITION TransGo:  END_TRANSITION	Uma transição pode ser definida graficamente usando FBD.

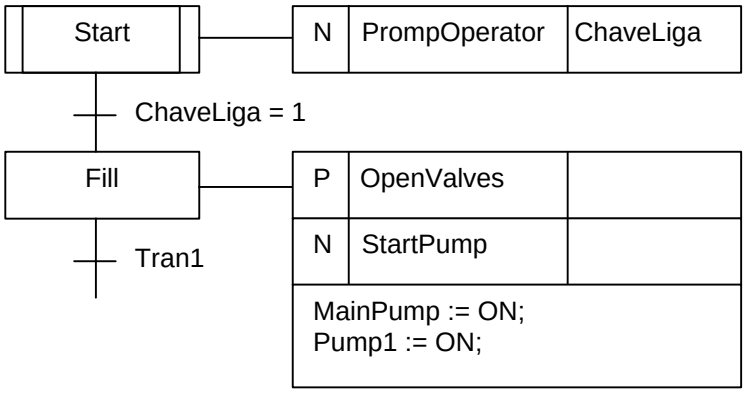
Transição definida usando LD	
TRANSITION TRANS21:  END_TRANSITION	A bobina deve ter o mesmo nome da transição. Quando a bobina é energizada, a condição de transição é verdadeira.

Transição definida usando IL	
TRANSITION Trans21: LD %IX21 AND EX10 AND FDIR21 END_TRANSITION	O último valor no acumulador corresponde ao valor a ser retornado. A operação deve retornar sempre um valor booleano. Se o valor for 1 o resultado será verdadeiro.

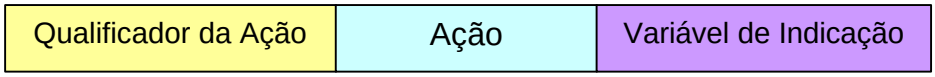
AÇÕES

Cada step pode ter uma mais ações associadas. Estas ações é que realizarão o trabalho de modificar o meio ambiente produzindo trabalho.

Cada ação pode ser descrita utilizando-se uma das quatro linguagens IEC 1131: ST, FBD, LD ou IL.



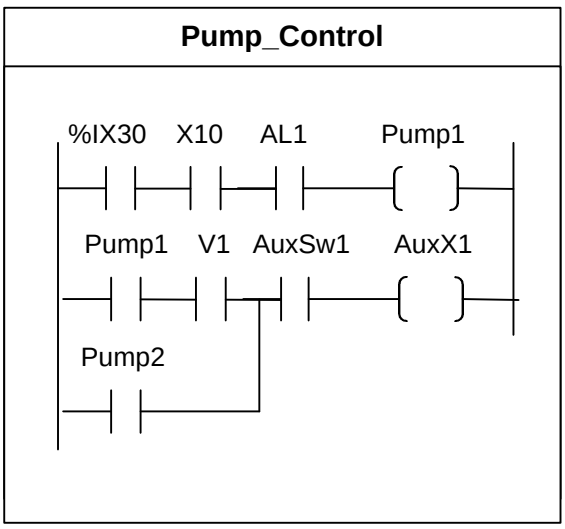
Cada ação tem um nome que deve ser único em cada POU.
As ações são representadas num retângulo contendo três campos.



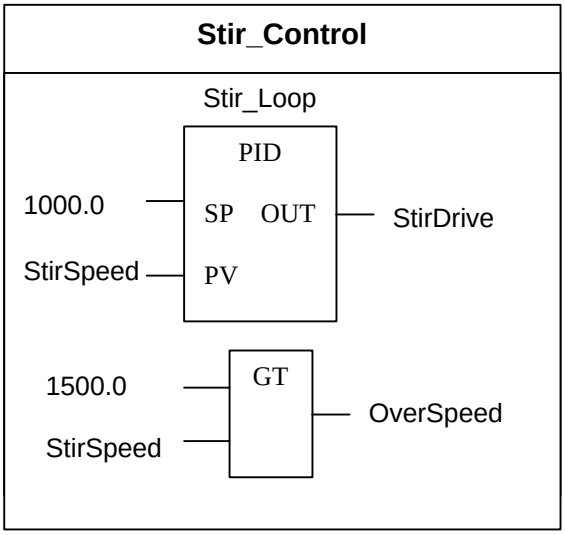
O primeiro campo à esquerda é o qualificador da ação. O qualificador determina quando a ação será executada.

Uma caixa de ação pode ter opcionalmente uma variável de indicação. A variável de indicação indica o nome de uma variável que é modificada dentro dado corpo da ação e que indica que a ação foi completada. A ação PrompOperator irá indicar através de StartSwitch que a função foi completada

Ações complexas podem ser definidas fora do SFC usando-se as linguagens LD, FBD, IL ou ST:

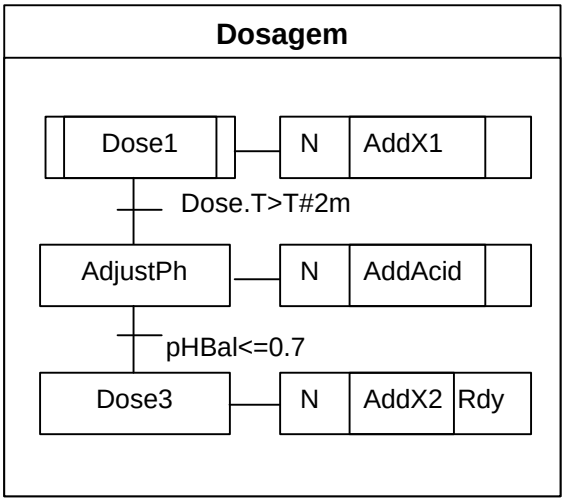
Ação definida através de LD	
	Os *rungs* são envoltos em uma caixa retangular com o nome da ação colocada no seu topo.

Ação definida através de FBD



A rede FBD deve ser circundada por um box retangular tendo o nome da ação em seu topo.
Se uma ação for definida para ser executada continuamente, ela pode conter loops PID ou outro tipo de loop realimentado de controle.

Ação definida através de SFC



Usando SFC podemos descrever ações complexas. Uma das variáveis de indicação de uma ação em SFC pode também ser usada como variável de indicação da ação do nível mais alto.

Para uma ação ser capaz de executar o seu próprio SFC, ela deve ser configurada para executar continuamente durante um período suficiente para permitir que seu SFC seja completado.

Ação definida através de ST	Uma variável modificada por uma diretiva ST pode ser usada como a variável indicadora da ação.

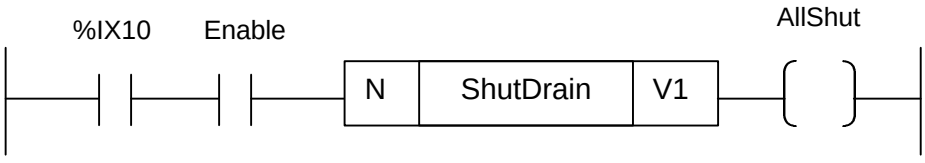
ACTION OpenValves:
IF BatchType = ÄBO_100" THEN
ValveAB := OPEN;
ELSE
ValveAC := OPEN;
END_IF;
BrainValve := CLOSE;
Vent := OPEN;
END_ACTION	

Ação definida através de IL	Uma variável modificada por uma instrução IL pode ser usada como a variável indicadora da ação.
ACTION AddX2: LD 1 SST TankX2.Enable LD 100 ST TankX2.ShotLevel CAL TankX2 S X2_added END_ACTION	

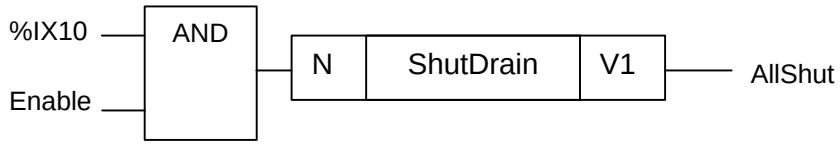
Uso de ações em linguagens gráficas:

Ações não têm seu uso restrito à SFC. Ações também podem ser usadas nas linguagens FBD e LD como no exemplo abaixo:

Exemplo Usando blocos de ação em diagramas ladder



Exemplo Usando blocos de ação em Diagramas de Blocos de Função



- Em FBD, a ação é ativada quando sua entrada booleana é TRUE.
- Em LD a ação é ativada quando existe um fluxo de potência dentro do bloco de ação.

Qualificadores de ações

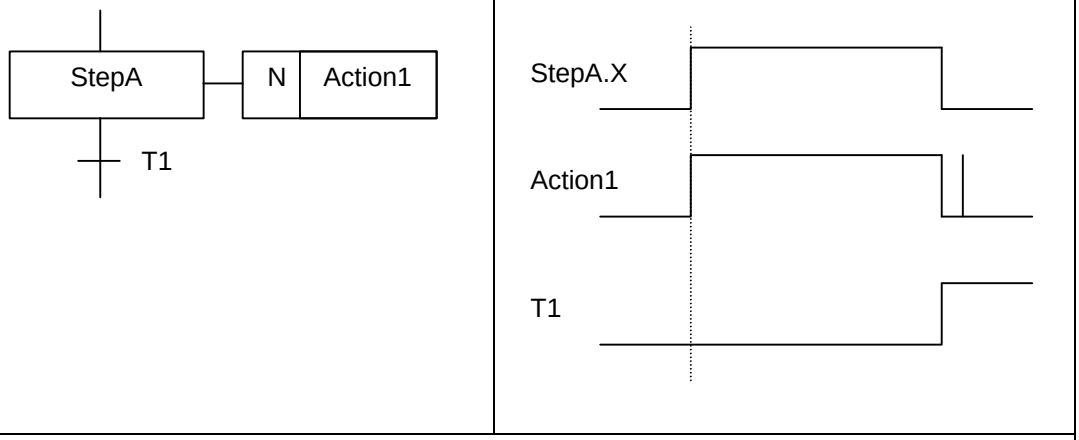
Qualificador	Descrição
Nenhum	Não armazenado, o mesmo que N
N	Não armazenado, executa enquanto o step associado estiver ativo
R	Reseta uma ação armazenada
S	Armazena uma ação ativa isto é armazenada. A ação continuará a ser executada até um qualificador R ser encontrado.
L	Ação limitada no temo, termina após um período estipulado
D	Ação adiada no tempo, começa após um período de tempo
P	Uma ação pulsada qua só é executada uma única vez quando o step é ativado e uma vez quando o step é desativado.
SD	Armazenada e com atraso de tempo. A ação é ativada após um tempo estipulado mesmo que o step associado for desativado antes do tempo de atraso.
DS	A ação é atrasa no tempo e armazenada. Se o step associado é desativado antes do período de atraso, a ação não é armazenada.
SL	Armazenada e limitada no tempo. A ação é iniciada e executada por um período de tempo.

Observação:

- Uma ação é dita armazenada quando continua após a desativação do estado, até que um qualificador R seja encontrado.

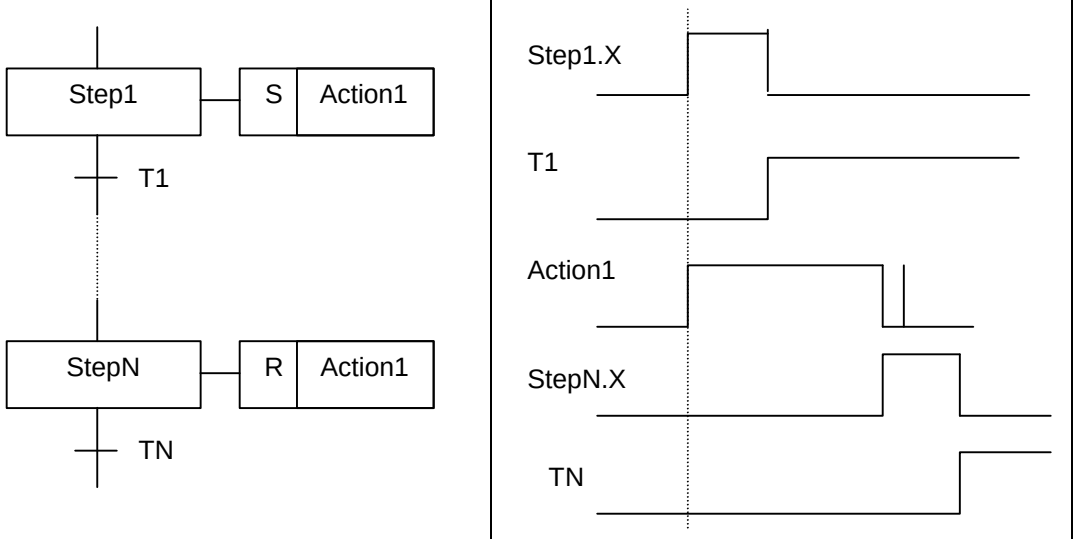
- Deve-se tomar cuidado com ações armazenadas que comecem muito tempo após seus estado terem sido desativados porque isto leva a programas confusos e de difícil depuração.

N - Ação não armazenada



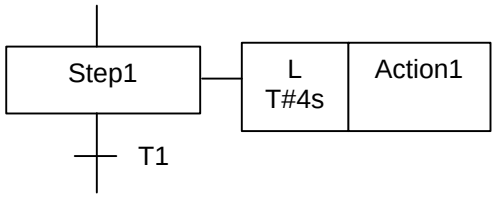
A ação executa continuamente enquanto o StepA está ativo, isto é enquanto a flag StepA.X = 1. A ação é executada uma última vez quando o step é desativado.

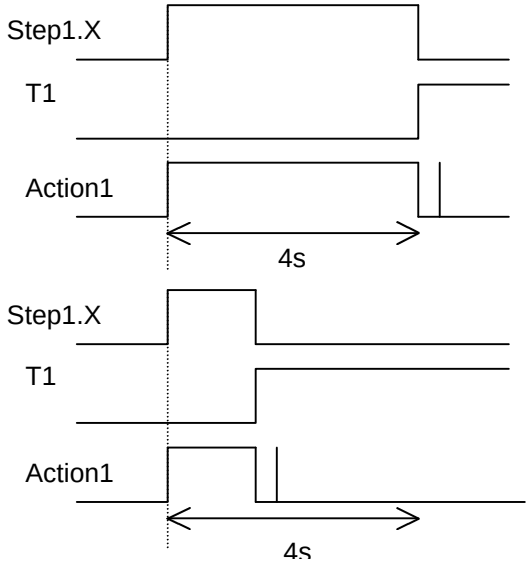
Set e **R**eset



A ação começa a ser executada quando StepA se torna ativo, e continua a ser executada até que o StepN é ativado. A ação associada ao estado StepN deve possuir o qualificador R e referenciar a mesma ação de nome Action1. A ação executa ainda uma última vez.

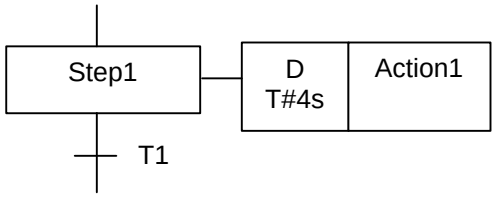
L - Ação limitada no tempo

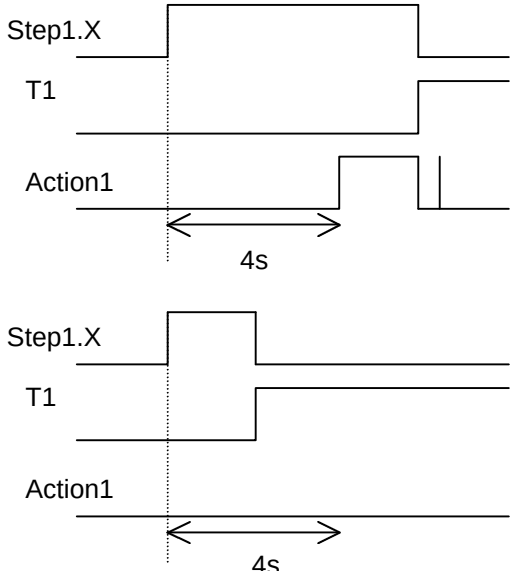




A ação executa durante o tempo estipulado a partir do instante de ativação do estado. Se o estado é desativado antes que o tempo haja expirado, a ação interrompe sua execução. A ação é executada uma última vez quando o step é desativado ou após o tempo haver expirado.

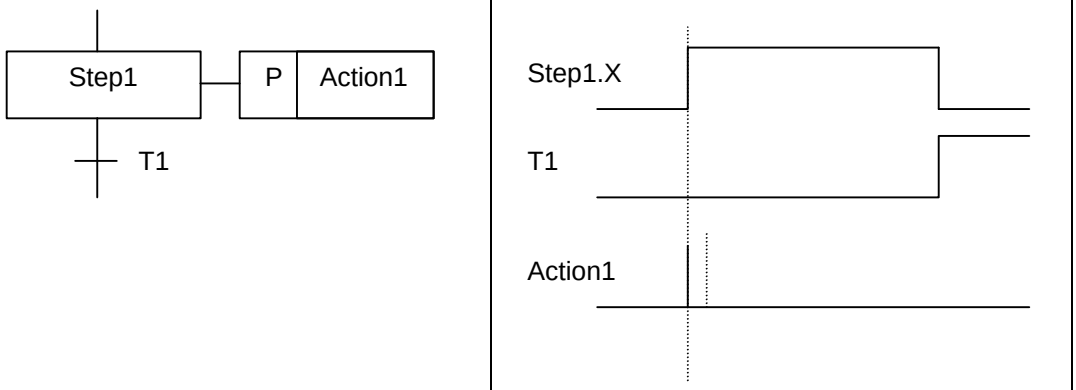
D - Ação com atraso de tempo





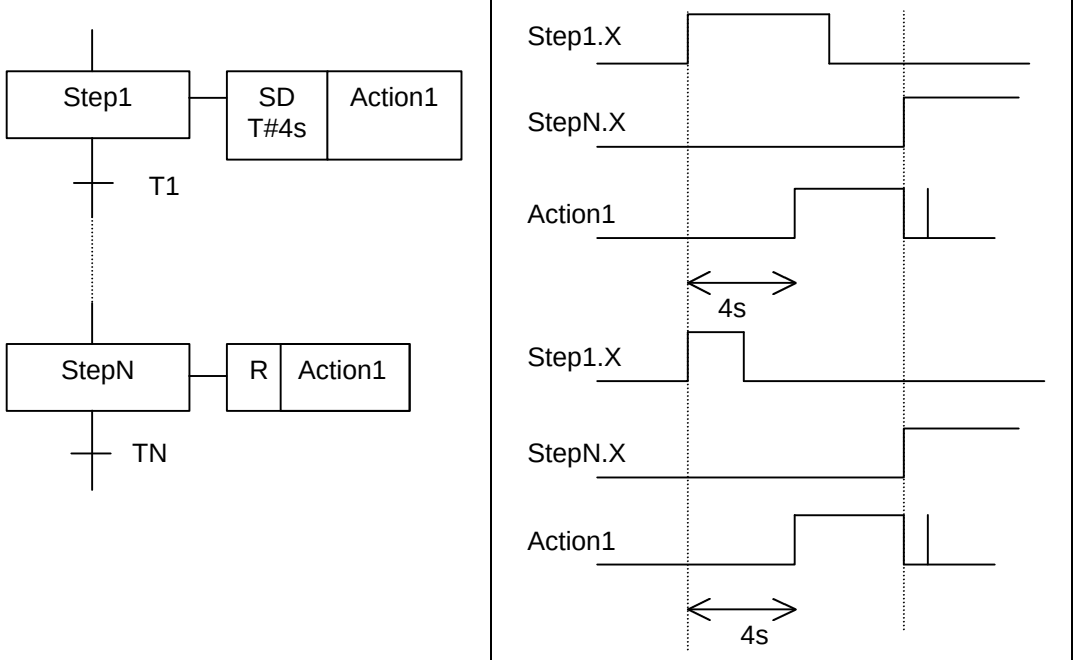
A ação executa após um atraso de tempo estipulado, a partir do instante de ativação do estado. Se o estado é desativado antes que o tempo de atraso haja expirado, a ação não é executada. A ação é executada uma última vez quando o step é desativado somente se tiver sido executada.

P - Ação pulsada



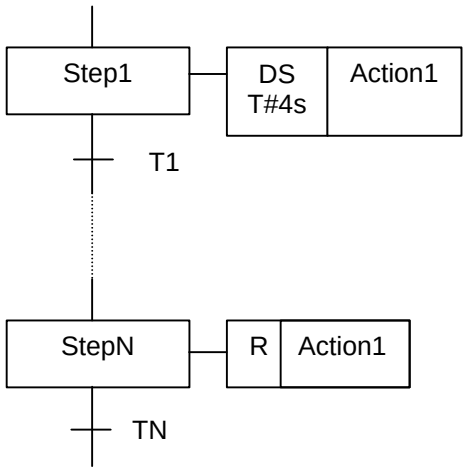
A ação é executada uma única vez após o estado ter sido ativado. Pelo padrão, a ação deve ser executada ainda uma segunda vez, entretanto isto irá depender da implementação.

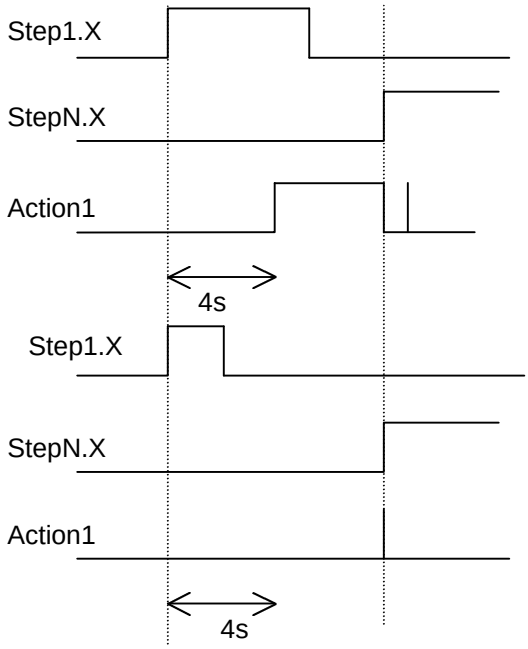
SD - Ação armazenada e adiada



Quando Step1 se torna ativo, a ação é armazenada, mas só começa a executar após o tempo ter se expirado. Independente da duração do Step1, a ação será executada até ser cancelada por uma ação com qualificador R referenciando a mesma ação (Action1). A ação não será executada se uma ação com qualificador R for executada antes do tempo de atraso expirar. A ação executa ainda uma última vez.

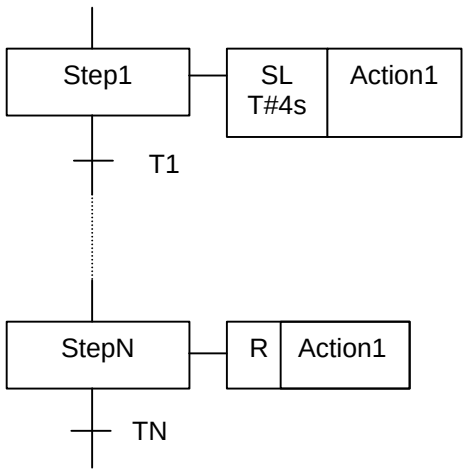
DS - Ação adiada e armazenada

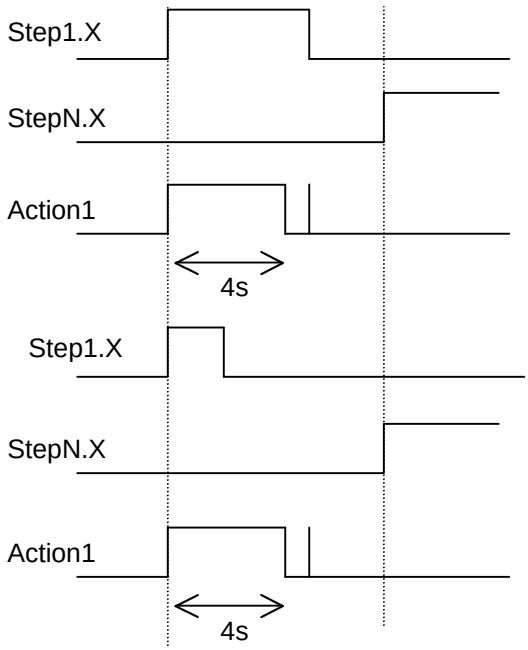




No começo do Step1 o tempo de atraso começa a ser computado. Após o tempo ter expirado, a ação é armazenada e começa a ser executada. A continua a ser executada até ser novamente referenciada por uma ação com o qualificador R. Se o Step1 for desativado antes do tempo expirar, a ação não será armazenada e não será executada.

SL - Ação armazenada e limitada no tempo





Quando o estado Step 1 se torna ativo, a ação é armazenada e começa a ser executada. Ela será executada até que o tempo T se esgote, independentemente do estado de Step1 e mais uma vez. Se a ação for desarmada por uma ação com qualificador R, antes do tempo se expirar, a ação será interrompida.

O bloco de funções controle de ação

Trata-se de um bloco de função conceitual definido no standard como meio de descrever o comportamento da ação. Cada ação é controlada por uma instância deste bloco. A saída do bloco de controle de ação é chamada Q . A ação associada ao bloco é executada, enquanto Q é verdadeiro. Todos os comandos da ação ou rede devem ser executados mais uma vez na borda de descida de Q. Ou seja, as ações são executadas uma vez extra.

E x e m p l o :

Suponha uma ação qualificada com o qualificador P e definida em ST:

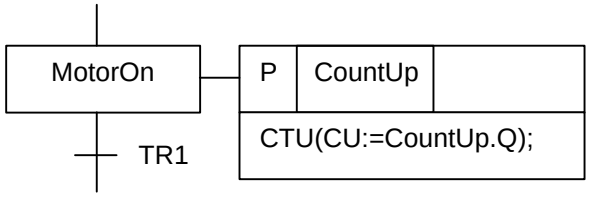
```
ACTION CONTING:
        COUNT := COUNT + 1;
END_ACTION
```

Assuma que esta ação está associada a uma step e se chama A1. Logo, toda vez que A1 é ativado, COUNT é incrementado de 1. Toda vez que A1 é desativado, COUNT é incrementado uma segunda vez !!!

Para executar um bloco uma única vez, devemos ter acesso ao qualificador da ação Q:

E x e m p l o :

Contar o número de vezes que um motor é acionado:



A ação CountUp é executada duas vezes. O contador entretanto só é incrementado na subida de CU.

Execução de um SFC

Redes SFC podem ser usadas para descrever o comportamento de uma POU ou de ações.

Cada POU está associada a uma task. A task executa periodicamente os elementos de uma POU. A SFC associada à POU será avaliada toda vez que a POU for executada. Uma SFC associada a uma ação será executada toda vez que a ação for executada.

Regras de avaliação de um SFC:

1. step inicial estará ativo quando a POU executa, após a inicialização do sistema. Quaisquer ações associadas com o passo inicial serão executadas.

2. No inicío de cada avaliação, o conjunto de steps ativos é determinado. Todas as transições associadas com os steps ativos serão avaliadas.

3. Ações que terminaram sua execução na última avaliação, são executadas mais uma vez.

4. Todas as ações ativas são executadas.

5. Todos os steps ativos que precedem transições avaliadas como verdadeiras são desativados e os seus estados sucessivos são ativados.

6. Todas as ações tendo uma condição de habilitação que abcabam de ser desativadas, são marcadas como inativas, por exemplo, uma ação com o qualificador de limite de tempo L, quando o período de tempo é esgotado.

Regras de evolução de um SFC:

O padrão IEC 1131-3 define um número de restrições na maneira como networks SFC são projetados e interpretados para que o resultado final seja livre de ambigüidade.

1. Dois steps não podem ser diretamente ligados. Eles devem ser separados por uma transição.

2. Duas transições nunca podem estar diretamente ligadas, elas devem ser sempre separadas por um step.

3. Se uma transição a partir de um step leva a dois ou mais steps, então estes steps iniciam seqüências simultâneas. Seqüências simultâneas continuam independentemente.

4. Quando projetando um SFC o tempo para realizar uma transição, desativar steps anteriores e ativar steps posteriores, deve ser considerado nulo.

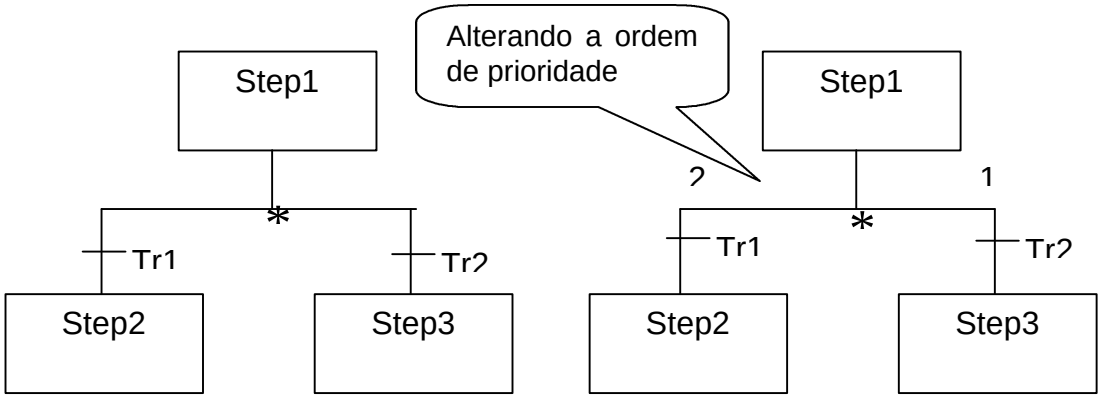
5. Quando mais do que uma condição de transição é verdadeira ao mesmo tempo, o tempo de ativação e desativação dos stps associados deve ser considerado nulo. O projetista não deve considerar as diferenças de tempo de transições simultâneas (condição de corrida).

6. A condição de transição de um step não é avaliada até que todo o comportamento resultante do step ativo tenha se propagado através da POU. Por exemplo, se um step foi ativado com a sua condição de saída sempre

TRUE, todas as ações serão executadas uma vez, antes do estado ser desativado.

Evolução de caminhos divergentes

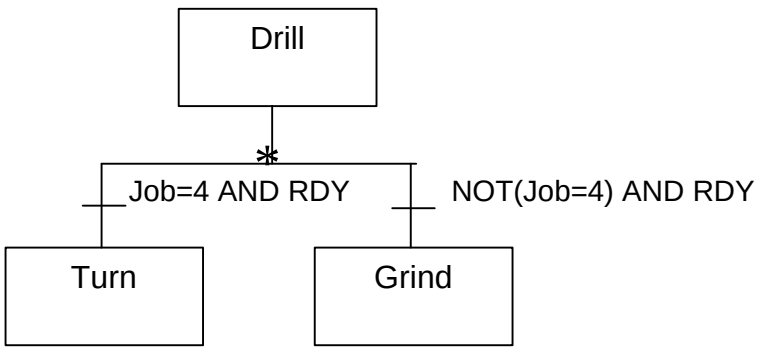
Quando existem duas ou mais transições a partir do mesmo step, existe ambigüidade sobre qual o caminho a ser seguido. Mesmo que várias transições estejam ativas simultaneamente, somente um caminho é selecionado. O caminho selecionado é determinado pela precedência da transição.



O default seria a avaliação da esquerda para a direita. Se as transições Tr1 e Tr2 forem ambas verdadeiras, a transição será feita para o Step2. A transição pela prioridade default é indicada no diagrama por uma asterisco colocado sobre o centro dos ramos divergentes.

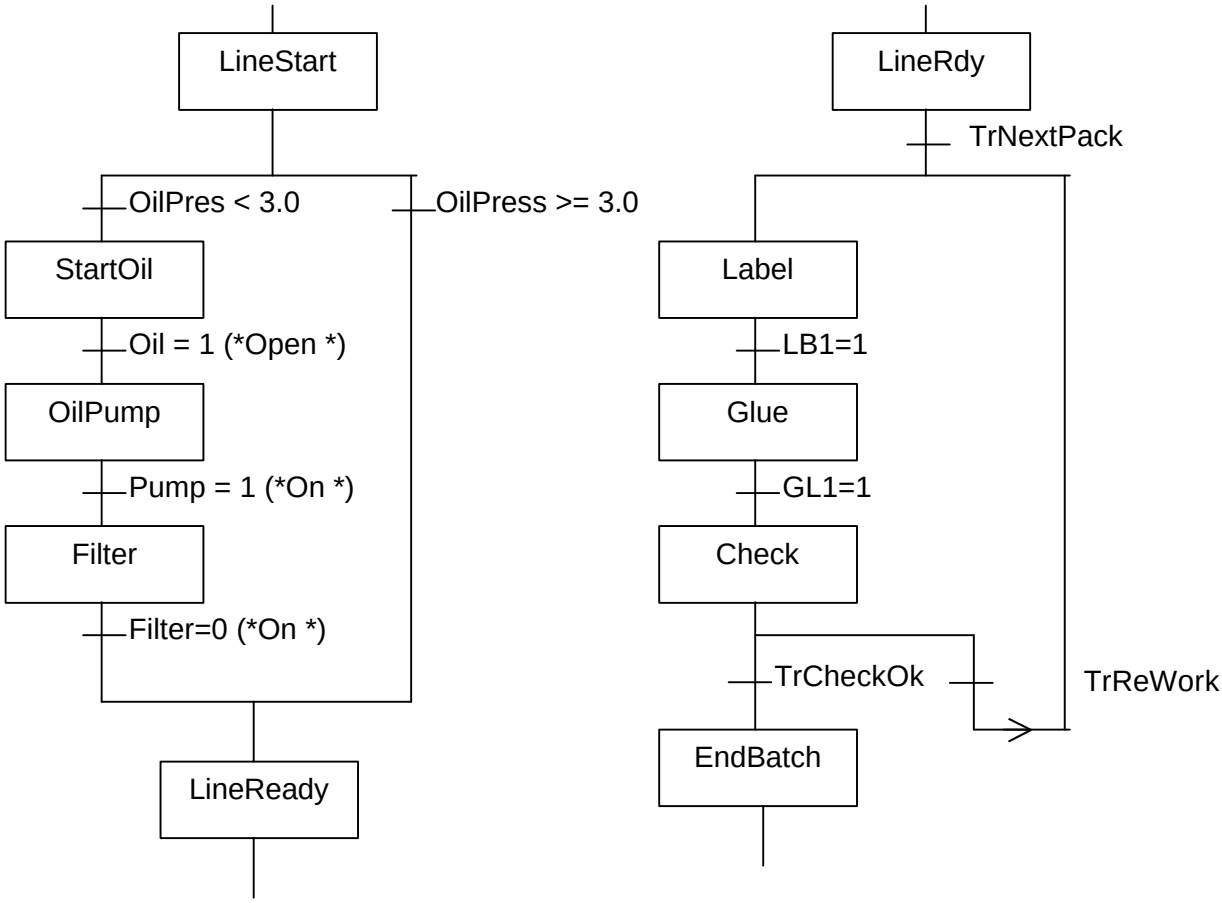
Caso desejamos uma outra ordem de precedência, devemos indicar a ordem desejada numerando os ramos divergentes e colocando o asterisco na junção.

A melhor maneira de se evitar ambigüidades é usar expressões mutuamente exclusivas para assegurar que apenas um dos ramos estará ativado. Esta alternativa tem como benefício extra estar conforme o padrão IEC 848.



Salto de seqüência

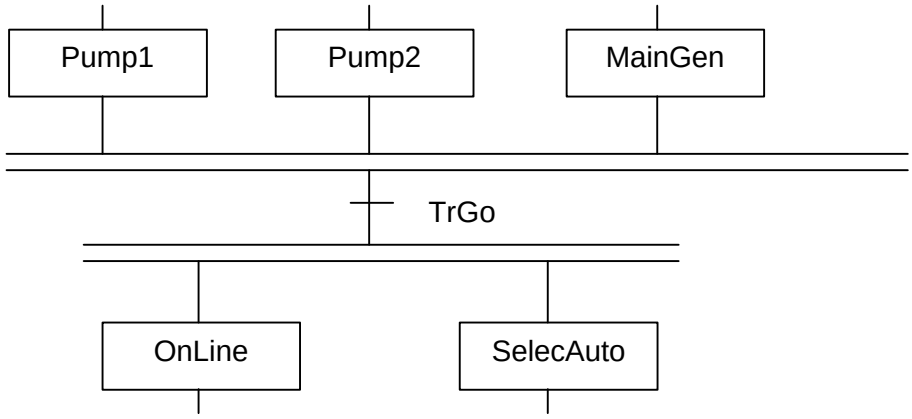
Pode-se usar um caminho divergente para saltar uma seção de uma seqüência. O caminho divergente não conterá steps e serve apenas para saltar obre o caminho alternativo.



Loop

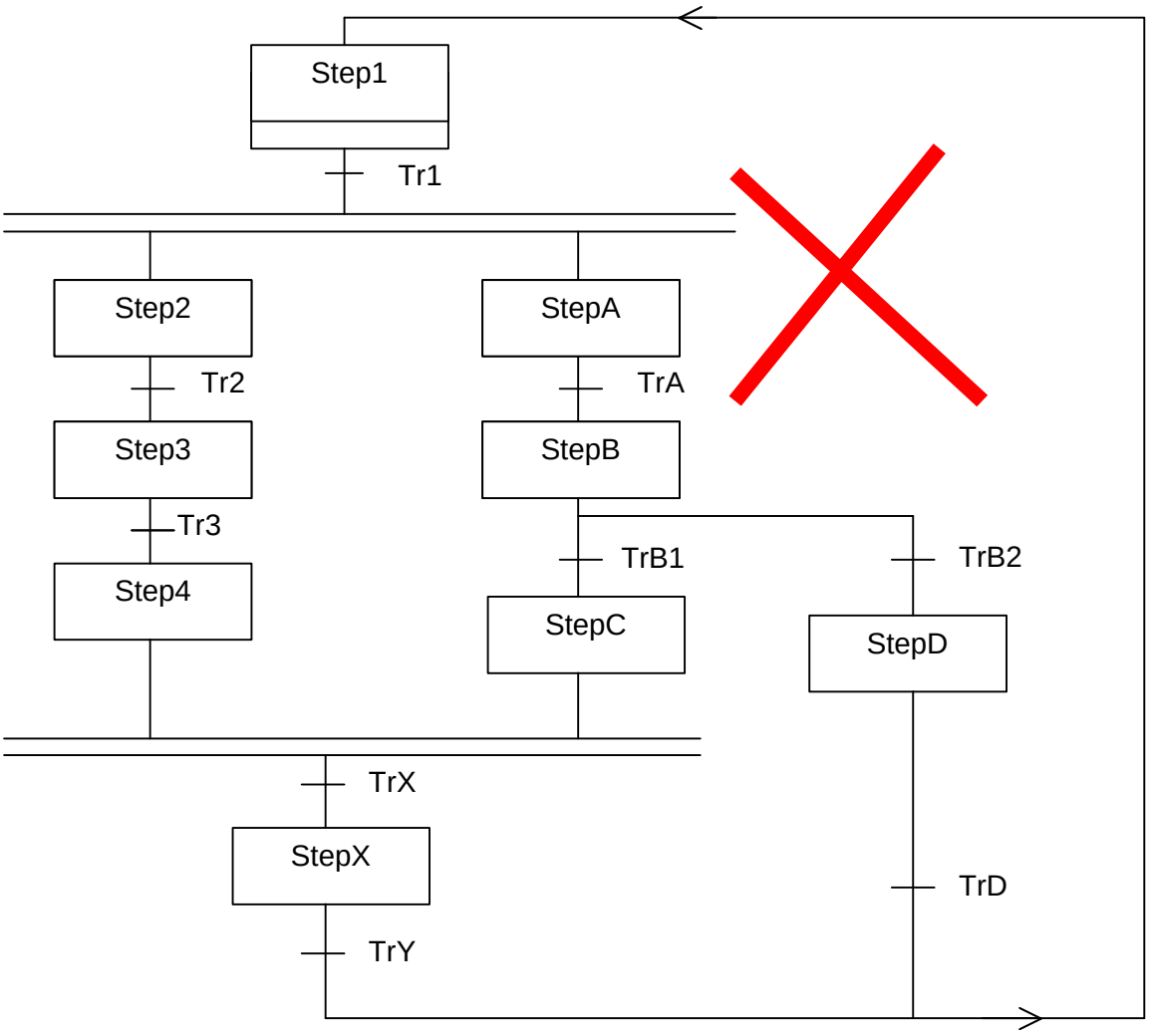
É permitido um salto para um step anterior, o que configura um loop.

Convergência Simultânea

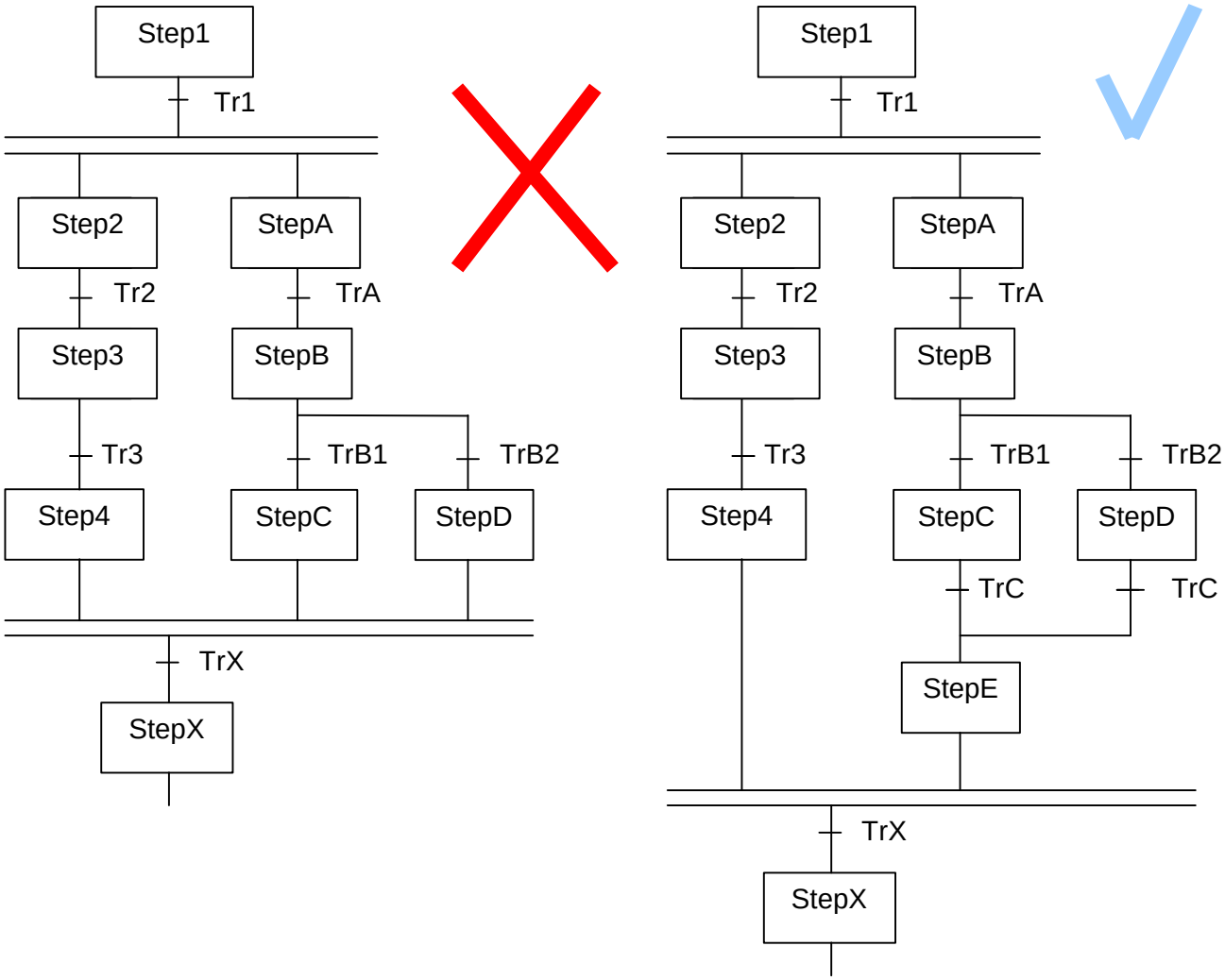


Na verdade serve apenas para sincronizar as sub seqüências. As seqüências devem se encontrar na convergência AND. Quando TrGo é disparado, as duas seqüências de saída são ativadas.

Projeto seguro e Inseguro



O diagrama acima é inseguro. Se o caminho por Step4 for tomado, poderá haver um retorno para o estado 1 com o a subseqüência Step2..Step4 ainda ativa. Se o SFC continua a ser executado, mais de um estado de uma mesma subseqüência a pode estar ativo ao mesmo tempo, o que conduz a um resultado imprevisível.



O SFC da esquerda tem um estado (StepX) que nunca será alcançado porque Steps C e D são mutualmente exclusivos, logo nunca os dois ramos serão percorridos simultaneamente.
O SFC da direita corrige esta situação introduzindo um estado intermediário (StepE).

Projeto Top down

Geralmente SFCs são utilizados para estabelecer a arquitetura de um sistema complexo. Primeiro os estados do sistema são identificados. Depois os cada estado do sistema é modelado através de um step do SFC principal.

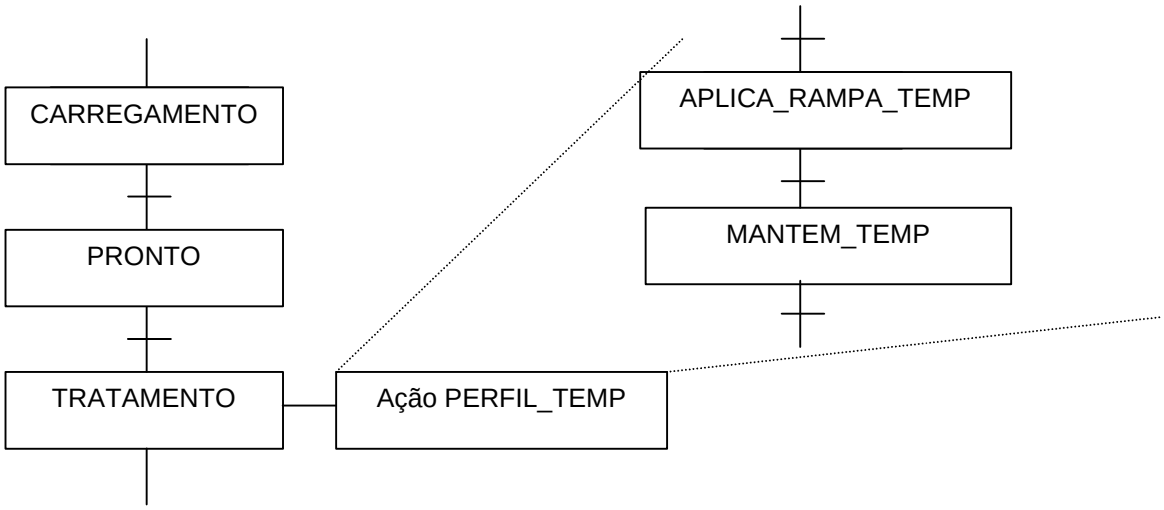
Por exemplo, se estivermos modelando um forno de reaquecimento de lingotes em uma siderúrgica, cada estado de funcionamento do forno é modelado como um Step do SFC:

Estados do Forno	
OCIOSO	O forno está frio e não está em uso
VAZIO	O forno não contém lingotes
AQUECENDO	O forno está seno aquecido para a temperatura de operação
PRONTO	O forno contém pelo menos um lingote e está com a porta fechada.
TRATAMENTO	O forno está carregado e na temperatura de operação
TERMINADO	O processo de tratamento do processo está completo e os lingotes podem ser descarregados.
RESFRIANDO	O forno está esfriando da sua temperatura de trabalho.

Cada estado poderia ser sub dividido novamente em sub estados. Por exemplo o estado TRATAMENTO poderia ser dividido em APLICA_RAMPA_TEMP, MANTEM_TEMP, etc. Esta nova seqüência pode estar contida dentro de outro SFC, por sua vez contido numa caixa de ação ou o SFC pode ser encapsulado em um bloco de função.

Depois de definidos os estados deve-se identificar quando as transições entre estados deve acontecer. Transições estão ligadas a eventos de operação (intervenções do operador) ou de processo.

SFC permite um enfoque *top down*, em que o comportamento do sistema é refinado sistematicamente, gerando SFC cada vez mais detalhados do comportamento do sistema.



Bom Estilo de Programação

1. Use nomes significativos para steps, transições e ações sempre que possível.
2. Tente manter cada SFC pequeno e focado no principal comportamento sendo considerado no nível hierárquico sendo analisado. Todo comportamento detalhado deve estar contido dentro dos blocos de ação e cobertos por SFCs de níveis inferiores.
3. Tente reduzir as interações entre seqüências simultâneas ao mínimo possível.
4. Evite que ações associadas aos steps de duas seqüências diferentes e simultâneas manipulem as mesmas variáveis, por exemplo parâmetros de entrada para blocos de função.
5. Cuidado deve ser tomado quando uma ação contém uma subseqüência eu pode ser paralisada em um estado intermediário por uma seqüência de nível superior. A seqüência de alto nível deve assegurar que todas as operações incompletas sejam limpas convenientemente.

Escopo de nomes dentro de redes SFC

Todos os nomes de estados, transições, conectores de transição e ações devem ser únicos dentro de uma POU, por exemplo bloco de função, que contém a rede SFC.

O escopo de um nome usado em um SFC dentro de um bloco de função é o bloco de função.

Elementos de SFC dentro de um bloco de ação pertencem ao mesmo escopo que as variáveis externas do SFC.

SFC em modo texto

SFCs podem ser definidos em modo texto. A utilidade seria ter uma notação canônica para transportar programas entre PLCs.

```
INITIAL_STEP Start:
        Prompt_Operator(N, StartSwitch);
END_STEP

TRANSITION FROM Start TO Fill:
        StartSwitch = 1;
END_TRANSITION
```

Step inicial

Transição:
Neste caso uma transição simples do estado Start para Fill

```
STEP FILL:
        OpenValves(P);
        StartPump(N);
END_STEP
```

Ações associadas ao step

```
ACTION StartPump:
        MainPump := ON;
        Pump1 := ON;
END_ACTION
```

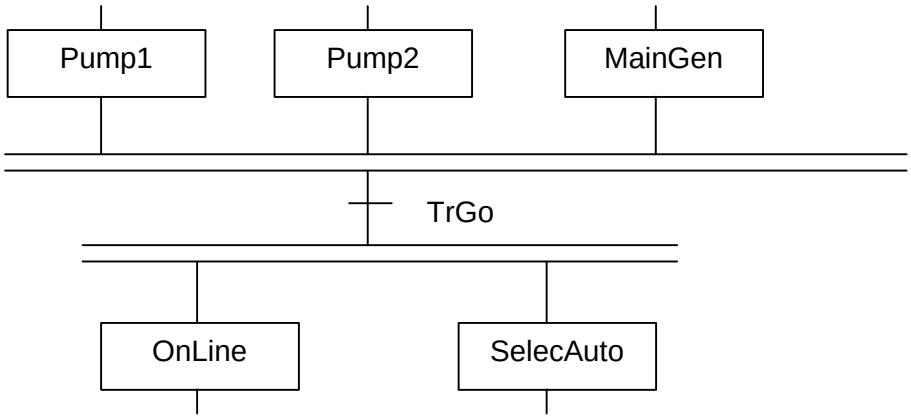
Comandos em ST ou IL

A ç ã o :

ActionName(Qualifier, Timed _qualifier, indicator variable);

E x e m p l o s :

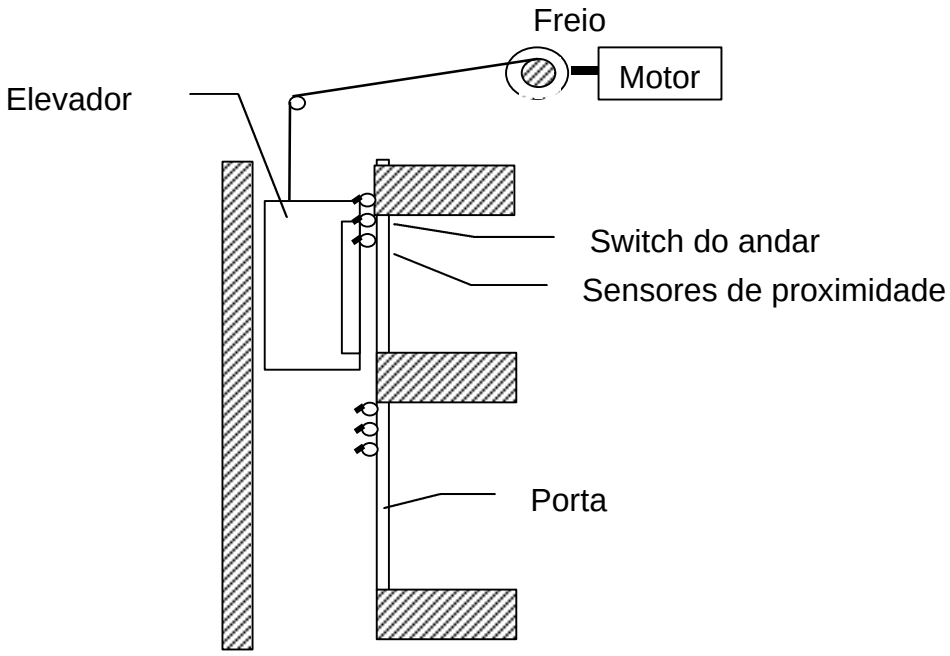
Action_StartUp(L, T#2m, Ready);
Action_Stop(P, Stopped);



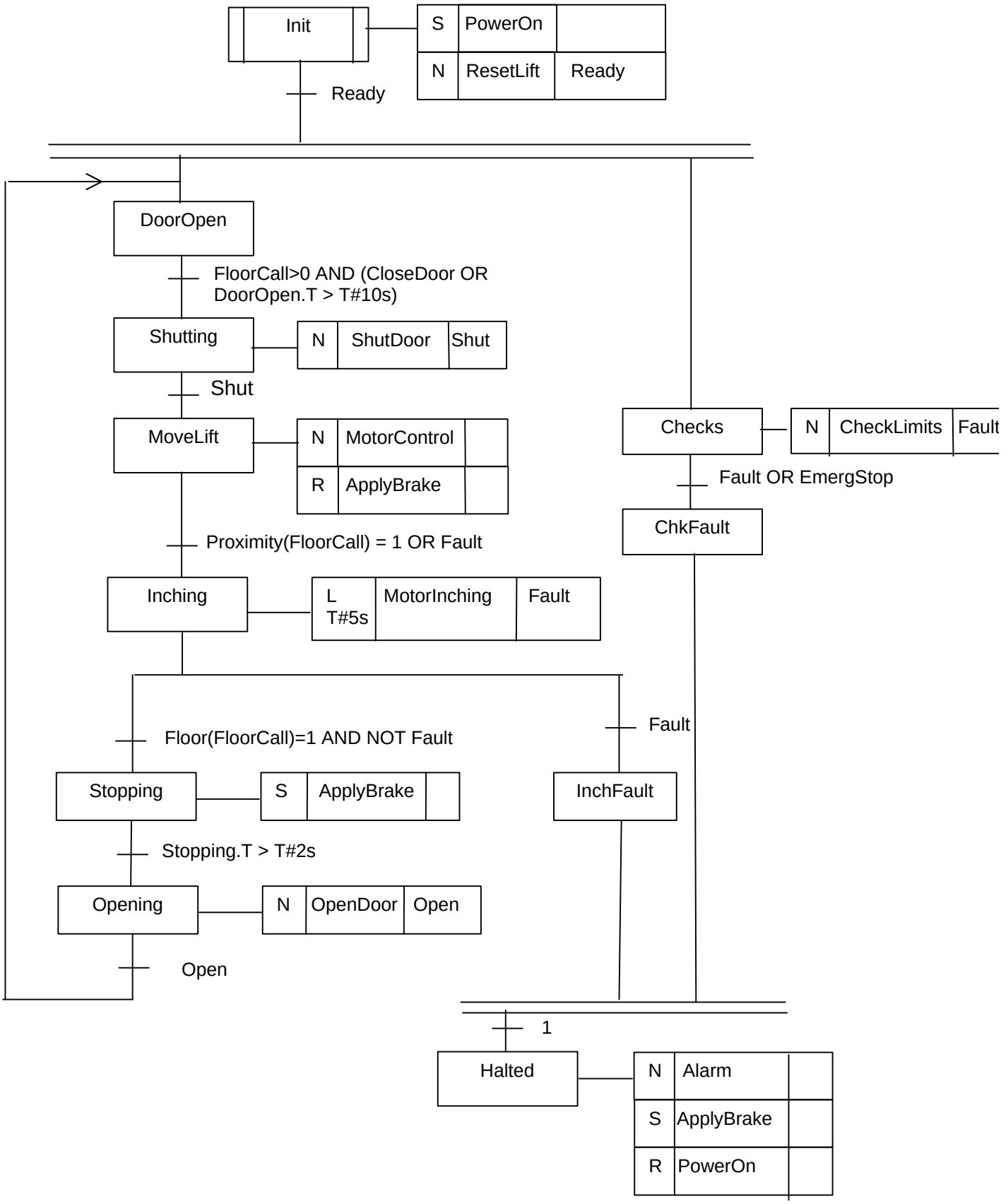
A transição TrGo colocada na forma canônica se torna:

```
TRANSITION FROM (Pump1, Pump2, MainGen) TO (OnLine, SelectAuto)
        := TrGo EQ 1;
END_TRANSITION
```

Exemplo: Controle de um elevador de carga



O elevador é puxado por um guincho e pode ser requisitado para parar em qualquer andar especificado.
Existem 3 chaves de proximidade: um na posição de parada do andar, outro acima e outro abaixo. A porta só se abre se a chave do andar estiver ativada.
O elevador deve desacelerar quando a primeira chave é acionada


Init
S
PowerOn
N
ResetLift
Ready
Ready
DoorOpen
FloorCall>0 AND (CloseDoor OR
DoorOpen.T > T#10s)
Shutting
N
ShutDoor
Shut
Shut
MoveLift
N
MotorControl
R
ApplyBrake
Proximity(FloorCall) = 1 OR Fault
Inching
L
T#5s
MotorInching
Fault
Floor(FloorCall)=1 AND NOT Fault
Stopping
S
ApplyBrake
Stopping.T > T#2s
Opening
N
OpenDoor
Open
Open
Checks
N
CheckLimits
Fault
Fault OR EmergStop
ChkFault
Fault
InchFault
1
Halted
N
Alarm
S
ApplyBrake
R
PowerOn

Leitura Complementar:

- Bonfatti, Monari, Sampieri, IEC1131-3 Programming Methodology, CJ International, 1997.
- Paul Barracos, Grafcet Step by Step - A tutorial and reference guide to the Grafcet automation Language - Famic, Canadá.

Exercícios:

1. Você quer definir expressões divergentes para evitar ambigüidade na avaliação de uma derivação OU. Qual das expressões abaixo são disjuntas (não intersecantes) com a expressão: $T = \overline{A}BC + AD + \overline{C}\overline{D}$?

   a) ( ) $T1 = A\overline{C} + C\overline{D}$
   b) ( ) $T1 = A\overline{C} + C\overline{D}$
   c) ( ) $T2 = A\overline{B}\overline{C} + CA\overline{D}$
   d) ( ) $T3 = \overline{C}$

2. Desenhe SFC para calcular o tempo médio de scan de um CLP avaliado a cada 1 segundo. O SFC deve também avaliar o tempo máximo e mínimo de scan já obtidos.

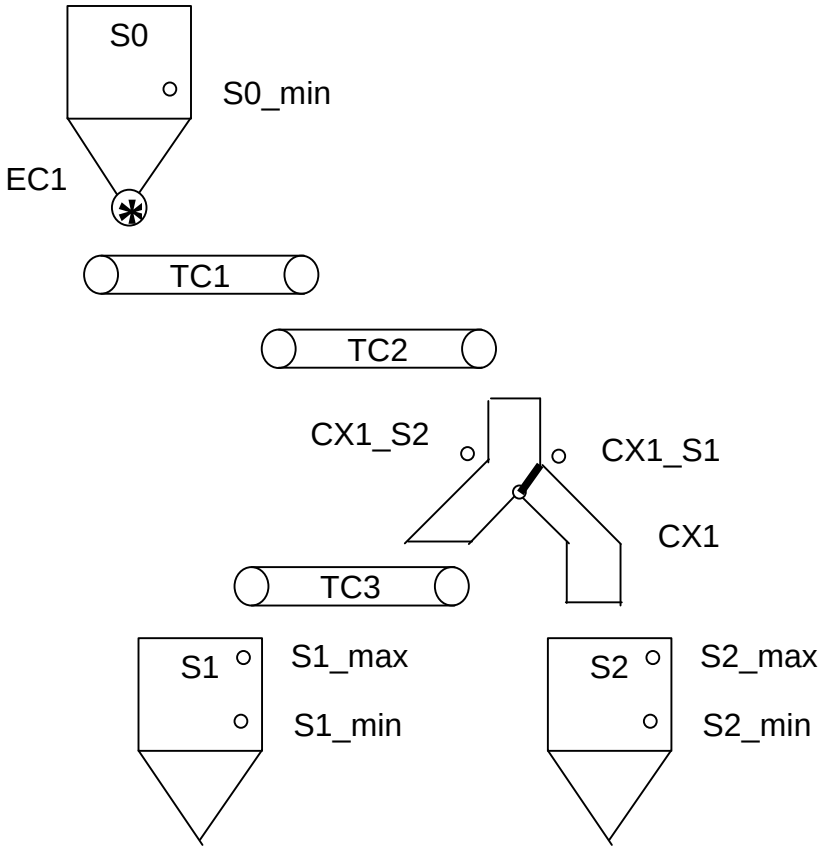
3. Carregamento automático de silos

   Deve-se automatizar o carregamento dos silos S1 e S2.

   - A eclusa EC1 do silo S0 despeja material na linha formada pelos transportadores TC1 e TC2,até que o detetor de nível alto do silo sendo carregado atue e a desligue. Ela será ligada quando o detetor de nível baixo de um dos silos atuar.
   - O desviador de fluxo é acionado toda vez que um dos silos estiver vazio. Caso ambos os silos estejam vazios (situação inicial), o silo S1 terá a preferência no enchimento. Os transportadores devem ser ligados na ordem correta a fim de alimentar os silos. O tempo de estabilização de rotação é de 15 segundos.
   - Existe uma chave de emergência que se acionada paralisa todos os equipamentos. Uma sirene deve ser disparada por 2 minutos. Uma nova partida só será autorizada se for pressionado o botão de rearme.
   - O número total de partidas comandadas, e o tempo de funcionamento das seqüências (S1 enchendo e S2 enchendo) devem ser computados e armazenados nas variáveis n_partidas e S1_func e S2_func respectivamente.

Relação de entradas/saídas:

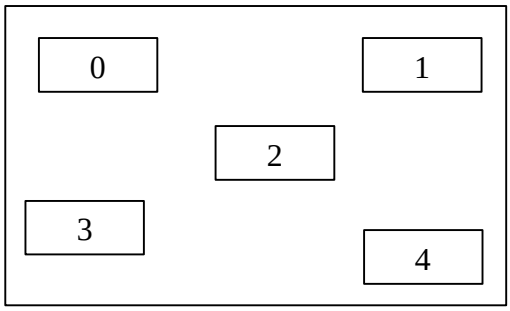
S0_min	detetor de nível mínimo do silo S0
S1_min	detetor de nível mínimo do silo S1
S2_min	detetor de nível mínimo do silo S2
S1_max	detetor de nível máximo do silo S1
S2_max	detetor de nível máximo do silo S2
CX1_s1	fim de curso do desviador CX1
CX1_s2	fim de curso do desviador CX1
EC1	Comando do motor da eclusa de descarregamento do silo S0
TC1	Comando do motor do transportador 1
TC2	Comando do motor do transportador 2
TC3	Comando do motor do transportador 3
CX1	Comando do Motor do desviador
CX1_dir	Direção do motor CX1: 0 = para silo S1



4. Uma tela tem 5 campos editáveis. Você deve editar os campos na ordem: 0, 3, 2, 1, 4. Você não pode sair da tela sem terminar a edição de todos os campos. Escreva o esqueleto de um programa que vá para o campo 1 e depois da edição de cada campo já dirija o cursor para o próximo campo a ser editado. Use autômatos de estados finitos como modelo de programação. A posição

X/Y de cada campo está no array PosicaoCampo. Para ir para o início de um novo campo use a instrução GotoXY(coluna, linha);

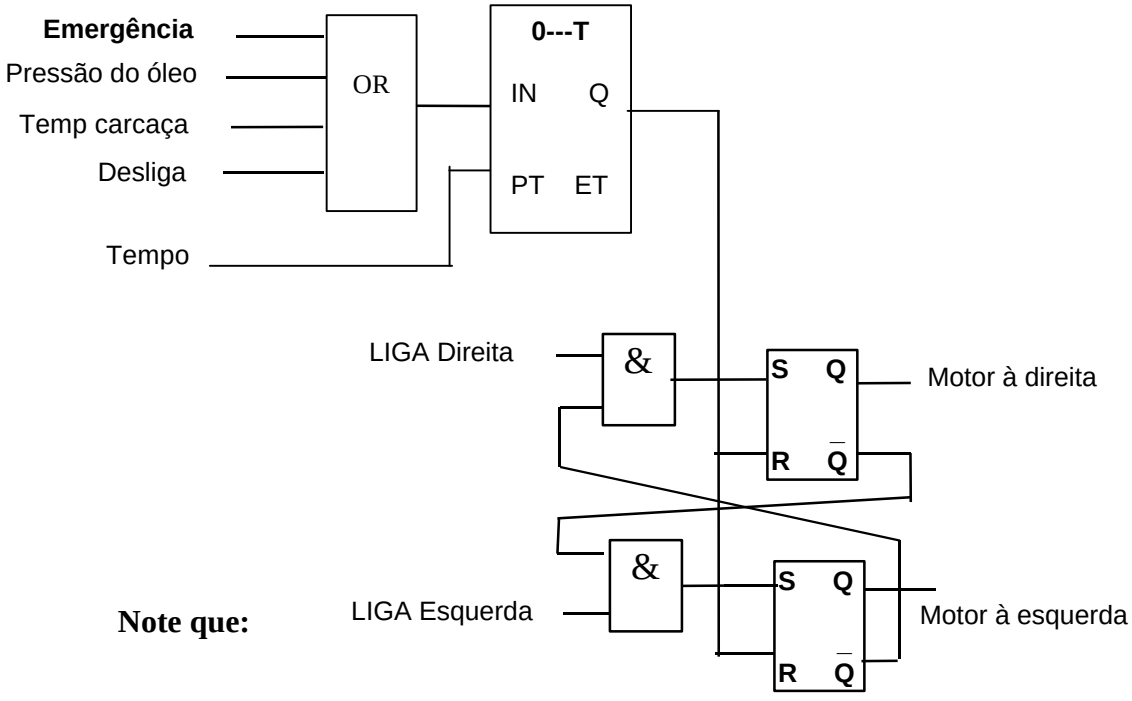
```
typedef struct {
    int       X;
int Y;
} Coord;
Coord PosicaoCampo[5]= {{0, 10}, {200, 10}, {50, 100}, {0, 100}, {0, 100}};
```



5. Motor reversível [Extraído de Automação Industrial - Ferdinando Natale - Nobel - Siemens series]

   Um motor elétrico pode ser acionado nos dois sentidos com reversão manual no sentido de rotação. O motor não deve entrar em funcionamento e deve ser desligado se qualquer uma das condições estiver presente:
   Se a chave de emergência for acionada.
   Se a pressão de óleo nos mancais for igual a zero.
   Se a temperatura da carcaça do motor for superior a 70°C.

   O diagrama lógico do sistema para ligar e desligar o motor é mostrado abaixo:



Note que:

- O temporizador com retardo no desligamento garante que sua saída ficará energizada durante o tempo em que a entrada se encontra energizada, e ainda durante um retardo de tempo T.
- O ligamento / desligamento do motor se dá via memórias biestáveis.
- Parar evitar que o motor que se encontra em um sentido de rotação seja ligado em sentido contrário, foi colocado um intertravamento no circuito.
- O desligamento deve ser único.

Construa a solução correspondente em SFC.

6. Descreva para que coisas você utilizaria cada uma das seguintes linguagens IEC61131-3:
   a) Ladder (LD)
   b) Lista de instruções (IL)
   c) Texto Estruturado (ST)
   d) Diagrama de blocos de funções (FBD)
   e) *Sequential Function Charts* (SFC)
   f) Fluxograma (ainda não padronizada).

   Algumas idéias:
   Intertravamento/ programação de batelada/ Motion Control/ Lógica de controle em indústria petroquímica/ lógica de proteção/ funções DCS / codificar blocos lógicos de alto desempenho /
   rotina fuzzy / algoritmos de otimização/ algoritmos genéticos/ partida de rotas de equipamentos, etc.