

Fundamentos de Automação

Licenciatura em Engenharia Electrotécnica

Lógica Programável – 2ª Parte



**ISEL
DEEA-SAR**

Armando Cordeiro



Lógica programável

- **Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET**
 - **GRAFCET (Gráfico Funcional de Comando Etapa Transição)**
 - Surgiu em França por volta dos anos 70;
 - Foi desenvolvido como uma metodologia baseada nas *redes de Petri*;
 - Inicialmente adoptado como IEC484 e actual IEC 60848 (1988);
 - Em 1993 passou a fazer parte da IEC 61131-3;
 - É um método gráfico que permite descrever, em forma de diagrama, as fases de funcionamento de um automatismo;
 - Descreve de uma forma clara, simples e de fácil compreensão o comportamento de um automatismo sequencial.



Lógica programável

- **Programação em GRAFCET**

- **Elementos gráficos do GRAFCET**

- **Etapas** – Mostram as diferentes fases do funcionamento do automatismo;
- **Transições** – Condições que fazem com que o processo evolua de uma etapa para outra;
- **Estrutura das ligações** – Define as ligações entre as etapas e transições da forma mais conveniente a produzir o resultado final. Por convenção, o sentido de evolução é de cima para baixo e é necessário indicação do sentido com uma seta;
- **Acções** – Correspondem às funcionalidades implementadas pelas etapas. Cada etapa pode efectuar diversas acções



Lógica programável

- **Programação em GRAFCET**

- **Regras de Evolução:**

- **Regra 1** – Duas etapas têm que estar sempre separadas por uma transição;
 - **Regra 2** – Duas transições têm que estar sempre separadas por uma etapa;
 - **Regra 3** – A transposição de uma transição só se dá quando ela for válida, ou seja, se todas as etapas anteriores a ela ligadas forem activas e a receptividade, isto é, a função lógica a ela associada, for verdadeira;
 - **Regra 4** – A transposição de uma transição provoca a desactivação de todas as etapas precedentes e a activação de todas as etapas imediatamente seguintes;
 - **Regra 5** – Uma transição pode dividir-se em duas ou mais etapas. Se tal acontecer, então são criadas sequências simultâneas, que independentes uma(s) da(s) outra(s);
 - **Regra 6** – Uma etapa pode dividir-se em duas ou mais transições. Se tal acontecer, então são criadas sequências alternativas.



Lógica programável

- **Programação em GRAFCET**

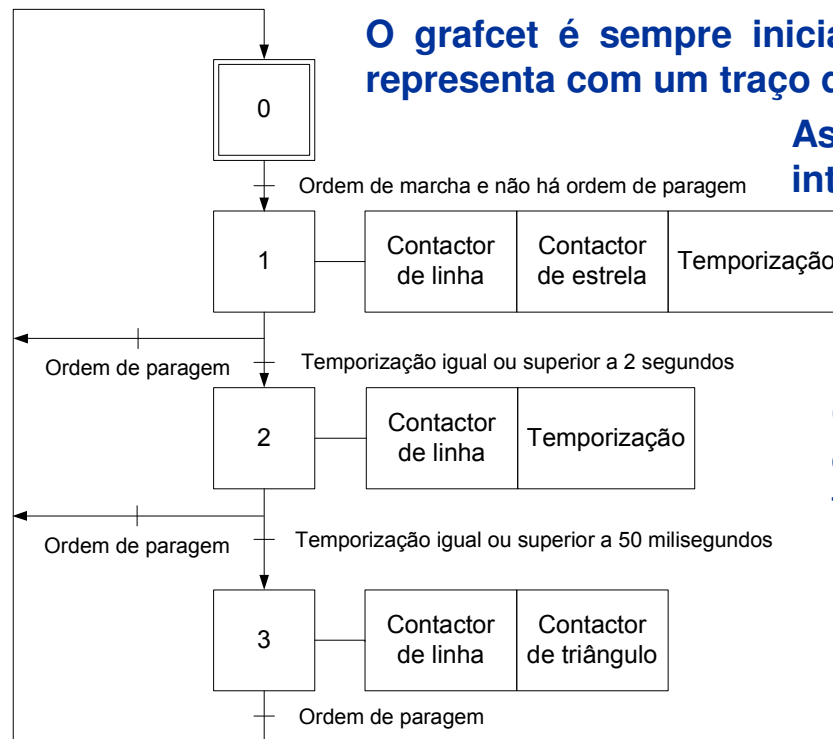
- **Como “desenhar” um Grafcet de um processo:**

- Todos os processos possuem, pelo menos, uma etapa inicial de repouso. Nesta etapa o autómato que controla o processo está ligado, mas o processo físico está parado, ainda não tendo existindo ordem, interna ou externa, para iniciar;
 - Quando existe uma ordem que obrigue o processo a iniciar o funcionamento, seja pelo operador, seja por instrução automática, deve-se impor ao grafcet uma transição, onde se reflita esta ordem, mudando-se de etapa;
 - Consoante as necessidades do processo, na etapa para a qual se transitou pode existir, ou não, uma, ou mais acções, a terem de ser realizadas. Considera-se que as acções são os accionamentos de saídas, temporizadores, contadores, memórias, relógios, operações matemáticas ou lógicas;
 - O grafcet irá permanecer nesta segunda etapa até existirem condições que o obriguem a transitar novamente para outra(s) etapa(s);
 - Quando se terminar o ciclo de funcionamento do processo, ou existir uma ordem de paragem, deve-se ter o cuidado de repor o grafcet na posição inicial de repouso, de modo que este fique a aguardar nova ordem de marcha.

Lógica programável

■ Programação em GRAFCET

➤ Exemplo com um arranque estrela-triângulo:



O grafcet é sempre iniciado com uma etapa de repouso (0) que se representa com um traço duplo

As transições refletem os eventos, externos ou internos, que obrigam o grafcet a evoluir

Após ordem de marcha é necessário ligar o contactor de linha e de estrela e temporizar o contactor de estrela

Garante-se o tempo de abertura do contactor de estrela antes de ligar o contactor de triângulo

Após a saída do contactor de estrela liga-se o contactor de triângulo, permanecendo-se nesta etapa, até existir ordem de paragem, retornando-se à etapa de repouso.

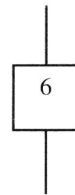
Sempre que exista uma ordem de paragem deve-se desligar o motor, bastando para tal sair das etapas onde os contactores estão activados, e retornar à etapa inicial de repouso.

Lógica programável

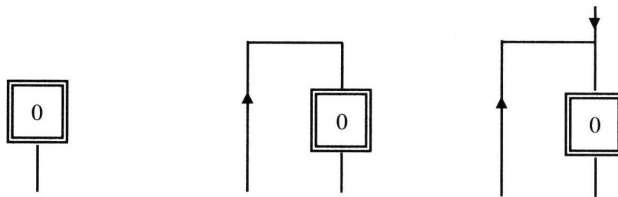
■ Programação em GRAFCET

➤ **Etapas** – correspondem aos estados do sistema e representam-se por:

■ Não Activas (E6=0)

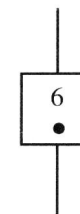


■ Iniciais

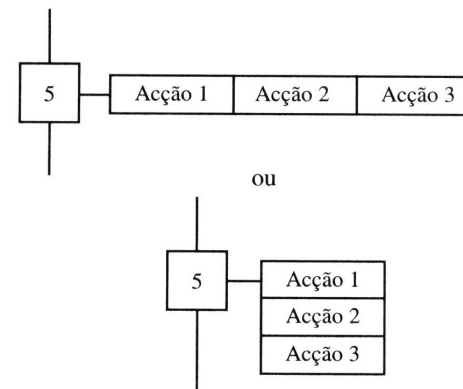


Etapa inicial sem retorno. Etapa inicial com retorno. Etapa inicial com retorno e com activação forçada.

■ Activas (E6=1)



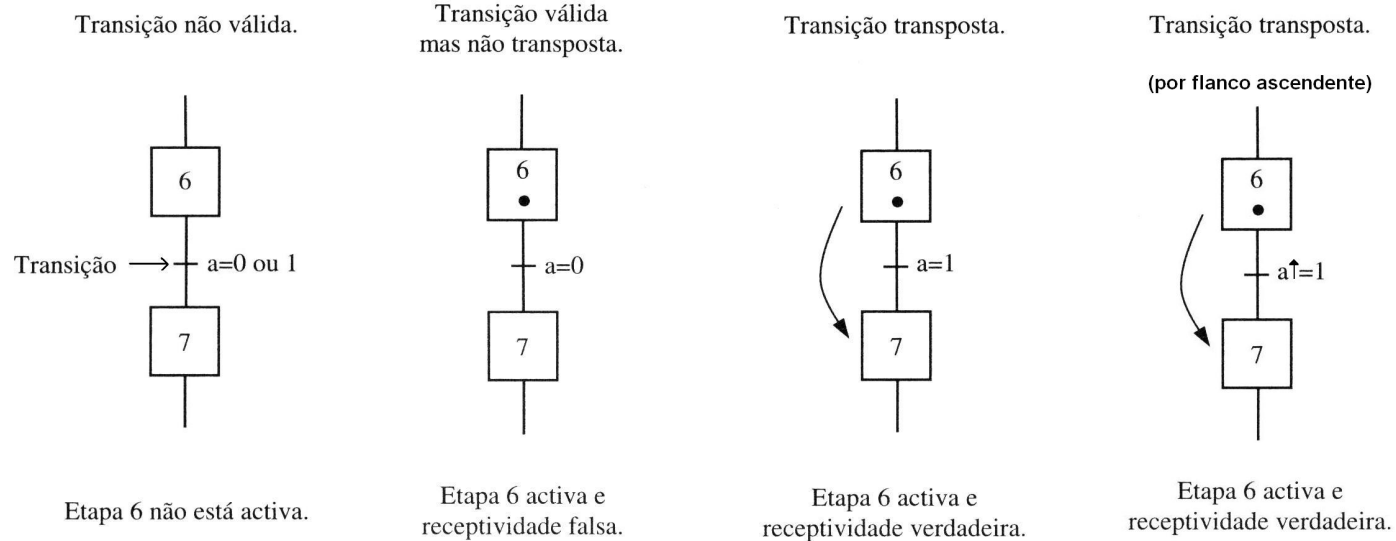
■ Acções associadas



Lógica programável

■ Programação em GRAFCET

- **Transições** – correspondem à possibilidade de passagem do estado activo de uma etapa para a seguinte e representam-se por:





Lógica programável

- **Programação em GRAFCET**

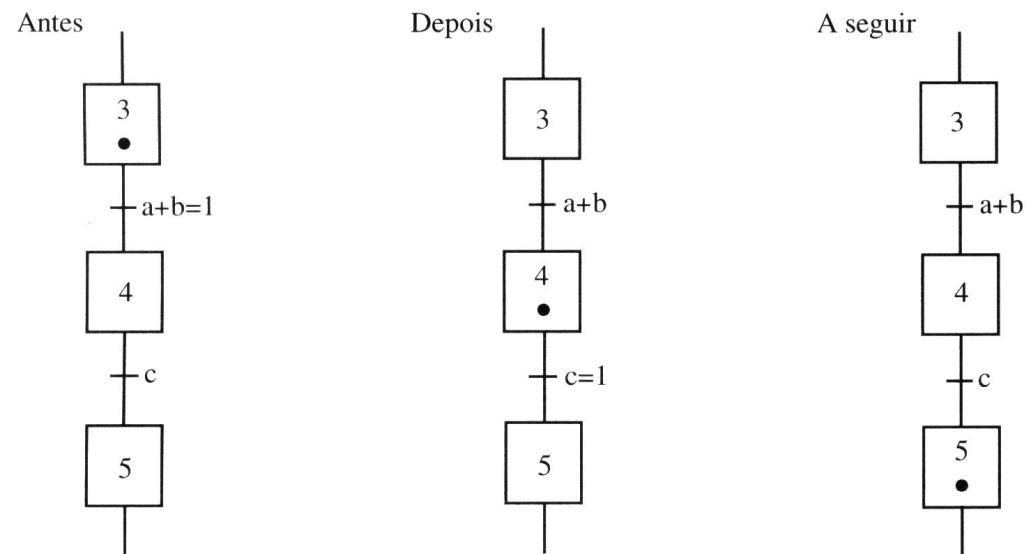
- **Tipos de estruturas de ligação entre etapas:**

- Sequência única
 - Sequências alternativas
 - Sequências simultâneas
 - Salto de etapas
 - Repetição de etapas
 - Outras estruturas de ligação

Lógica programável

■ Programação em GRAFCET

➤ Estruturas de ligação – Sequência única

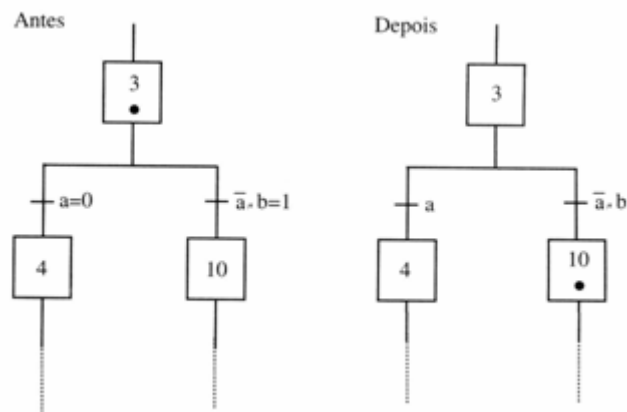


Lógica programável

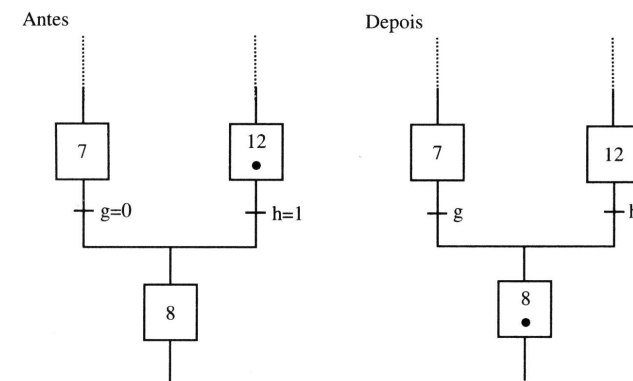
■ Programação em GRAFCET

➤ Estruturas de ligação – Sequências alternativas

■ Divergência OU



■ Convergência OU



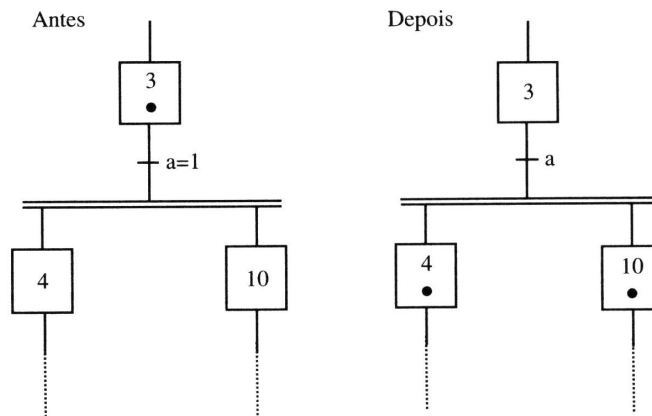
As receptividades das transições de sequências alternativas devem ser mutuamente exclusivas, de modo a activar apenas um dos ramos da sequência.

Lógica programável

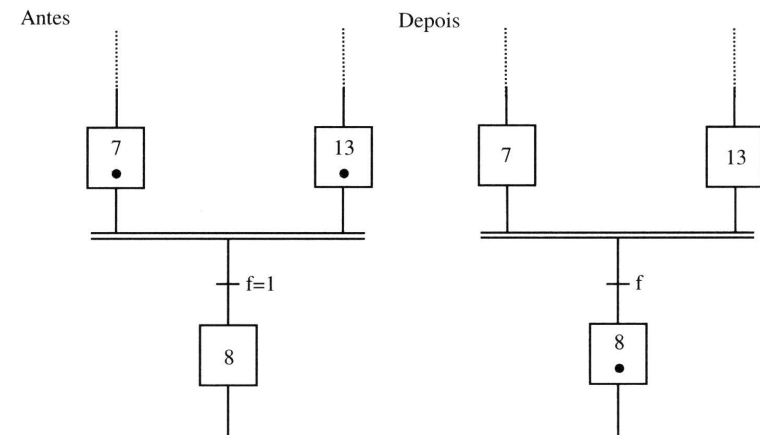
■ Programação em GRAFCET

➤ Estruturas de ligação – Sequências simultâneas

■ Divergência E



■ Convergência E



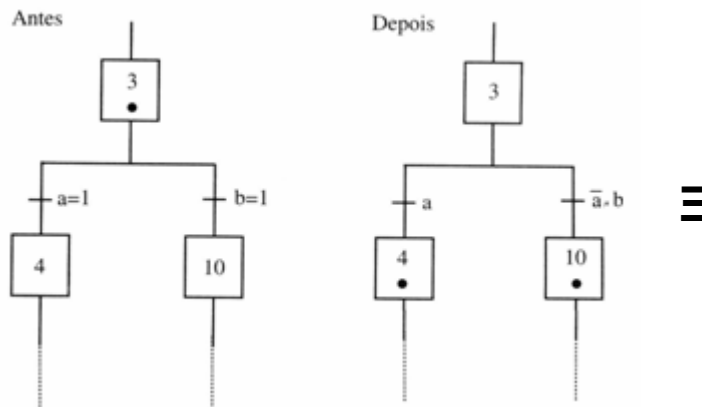
➤ É frequente que o tempo de execução de cada uma das sequências simultâneas seja diferente e como a convergência E só pode ser ultrapassada quando todas as etapas terminais estiverem activas é necessário introduzir etapas suplementares de espera.

Lógica programável

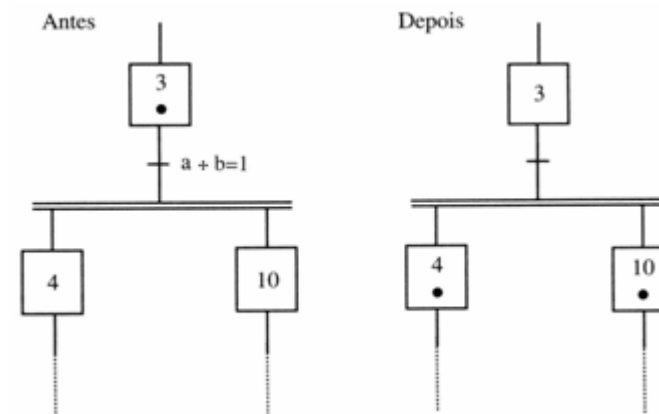
■ Programação em GRAFCET

➤ Estruturas de ligação – Sequências alternativas/simultâneas

■ Divergência OU



■ Divergência E



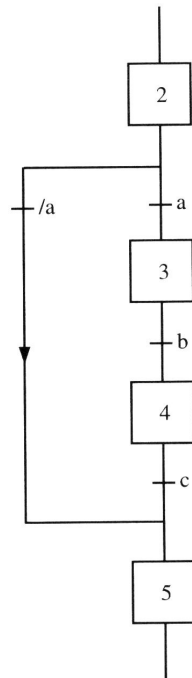
- Quando as receptividades das transições de sequências alternativas **não são mutuamente exclusivas**, pode-se dar o caso de se activar os dois ramos da sequência, originando um comportamento idêntico ao da sequência simultânea.

Lógica programável

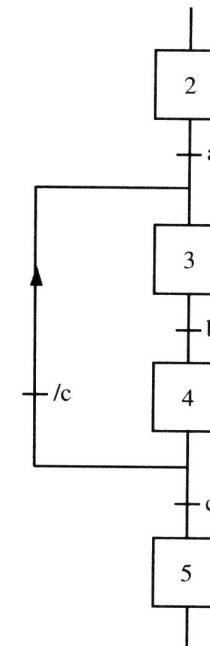
■ Programação em GRAFCET

➤ Estruturas de ligação – Salto e repetição de etapas

■ Salto de etapas



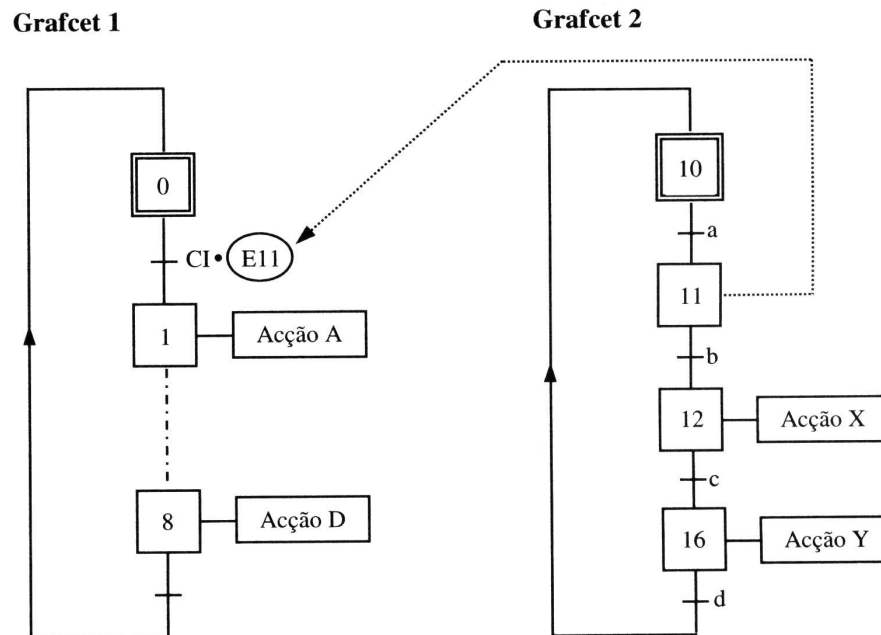
■ Repetição de etapas



Lógica programável

■ Programação em GRAFCET

➤ Estruturas de ligação – Outras estruturas (Diagramas ligados)



➤ Neste caso a ligação entre graficets é feita pela afectação da transição com o bit de uma etapa de outro graficet, não existindo uma verdadeira ligação física, mas também pode existir ligação pelo simples facto de se utilizarem os mesmos sensores que monitorizam o processo.

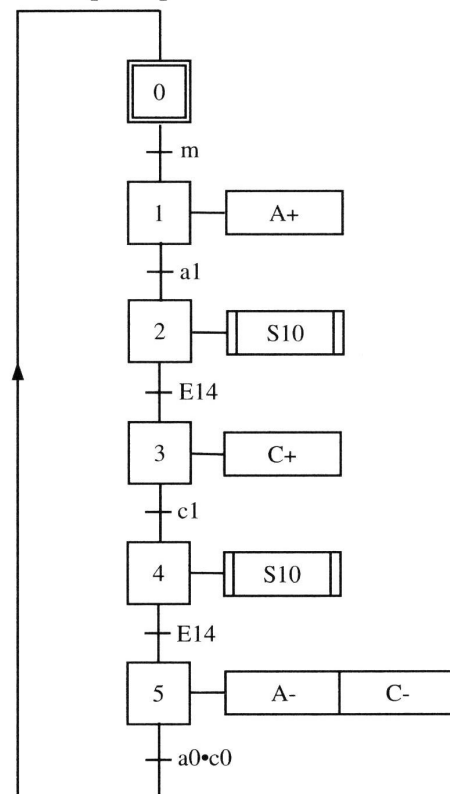
➤ Um exemplo típico desta estrutura é a utilização de um graficet para o controlo do processo e de um segundo graficet para o controlo das paragens de emergência.

Lógica programável

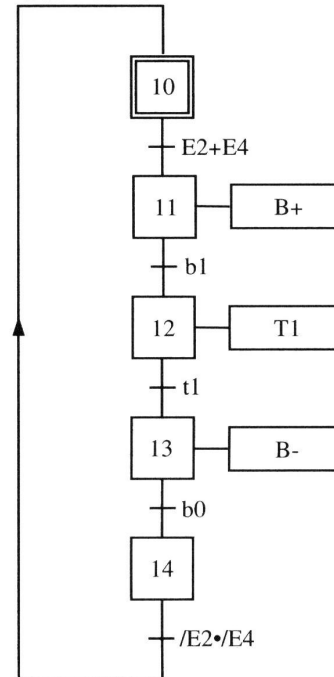
■ Programação em GRAFCET

➤ Estruturas de ligação – Outras estruturas (Subrotinas)

Grafcet principal



Sub-rotina



➤ Este caso difere do anterior na medida em que a acção de uma etapa do grafcet principal é a execução de um outro grafcet.

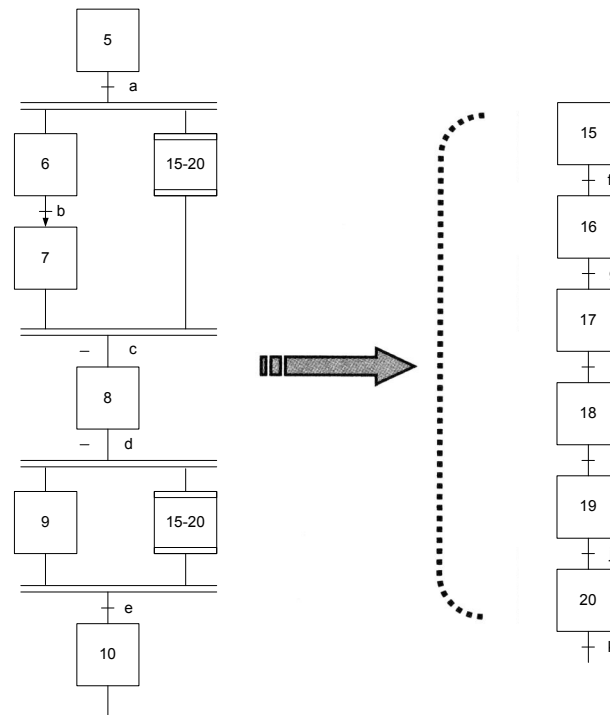
➤ Se a acção da etapa for muito complexa é possível utilizar-se sub-rotinas de modo a simplificar-se o grafcet principal.

Lógica programável

■ Programação em GRAFCET

➤ Estruturas de ligação – Outras estruturas (Macroetapas)

- Destinam-se a simplificar e a tornar mais legíveis os diagramas do GRAFCET.



➤ Quando os graficets possuem um determinado troço de sequência que se repete é possível simplificar através de macroetapas.

Lógica programável

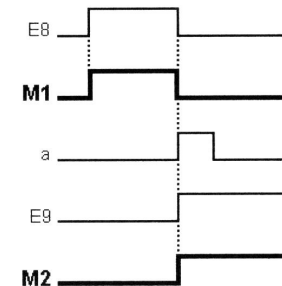
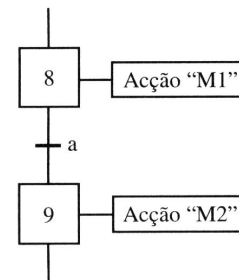
■ Programação em GRAFCET

➤ Tipos de Acções

■ Tabela de acções possíveis

Letra	Acção
N ou nenhuma	Não memorizada
S	Memorizada
R	Colocar a zero
L	Limitada no tempo
D	Temporizada
C	Condicionada
P	Impulso
SD	Memorizada e temporizada
DS	Temporizada e memorizada
SL	Memorizada e limitada no tempo

■ Acções não memorizadas

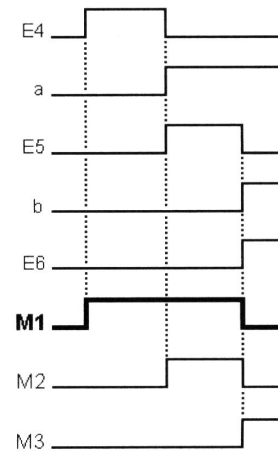
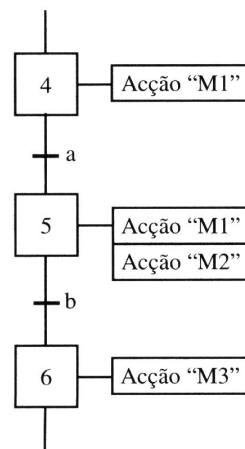


Lógica programável

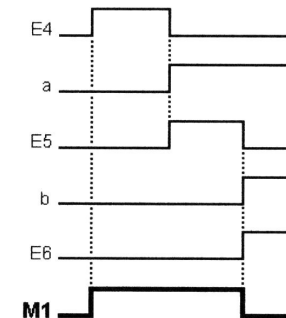
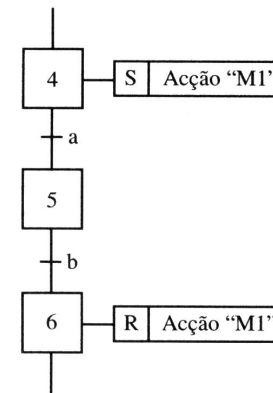
■ Programação em GRAFCET

➤ Tipos de Acções

■ Acções memorizadas



■ Acções memorizadas (S/R)

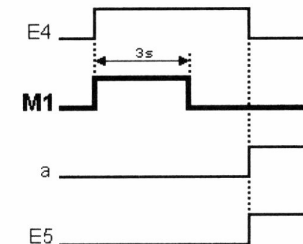
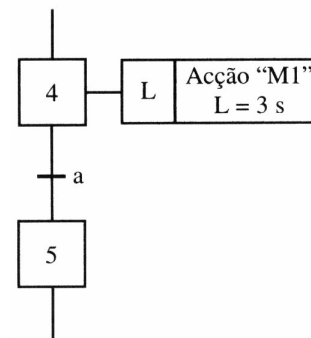
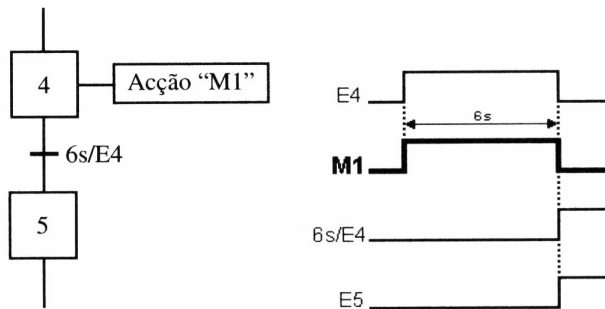


Lógica programável

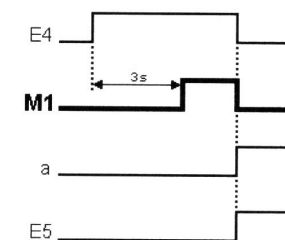
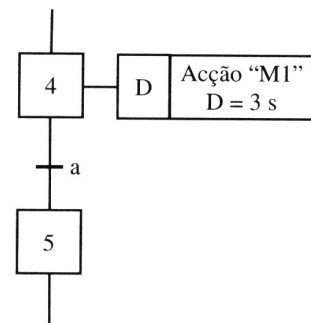
■ Programação em GRAFCET

➤ Tipos de Acções

■ Acções dependentes do tempo



- Transição temporizada o que origina acção activa durante um determinado período

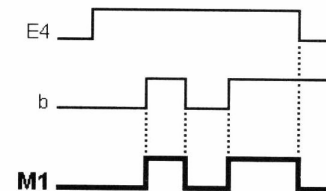
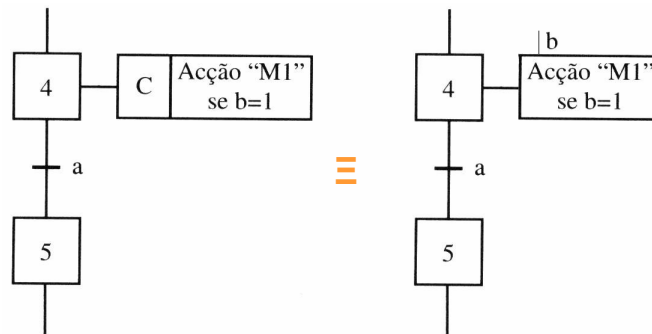


Lógica programável

■ Programação em GRAFCET

➤ Tipos de Acções

■ Acções condicionadas



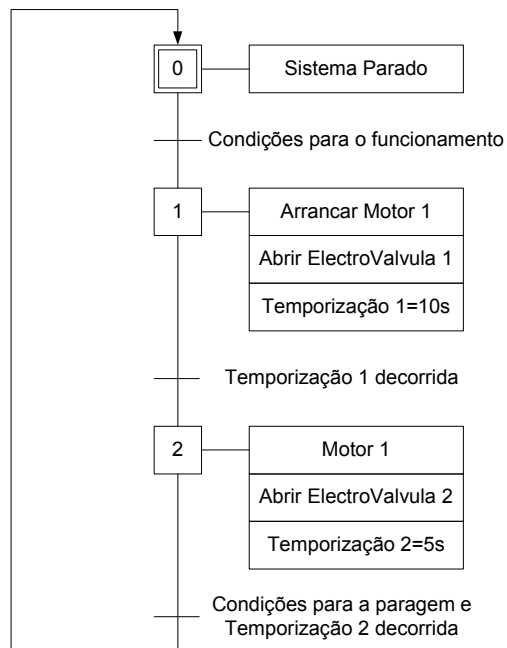
- A acção só é realizada quando a etapa estiver activa e a condição b for verdadeira

Lógica programável

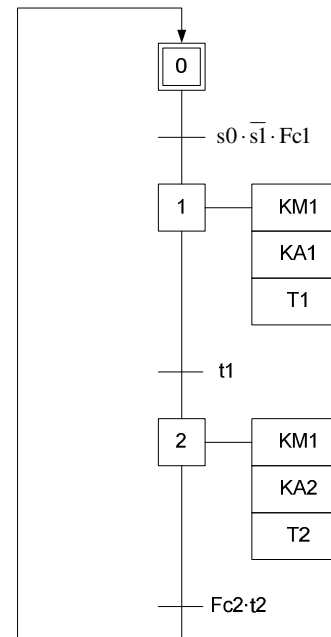
■ Programação em GRAFCET

➤ Níveis de Grafcet – Exemplo básico

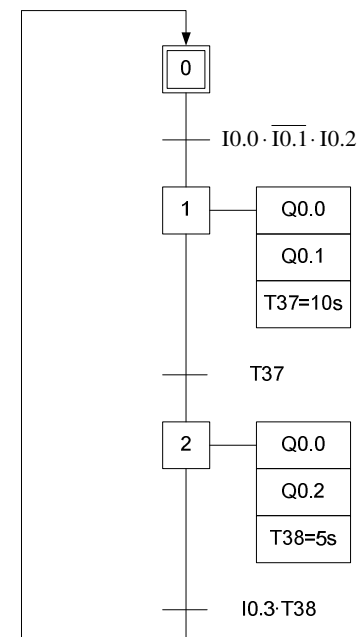
Especificações funcionais (Nível 1)



Especificações tecnológicas (Nível 2)



Especificações operacionais (Nível 3)

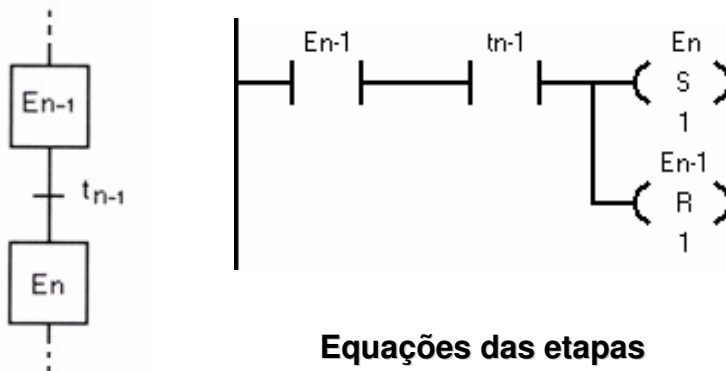


Lógica programável

■ Programação em GRAFCET

➤ Metodologia I para programar o grafcet

Caso Genérico



Equações das etapas

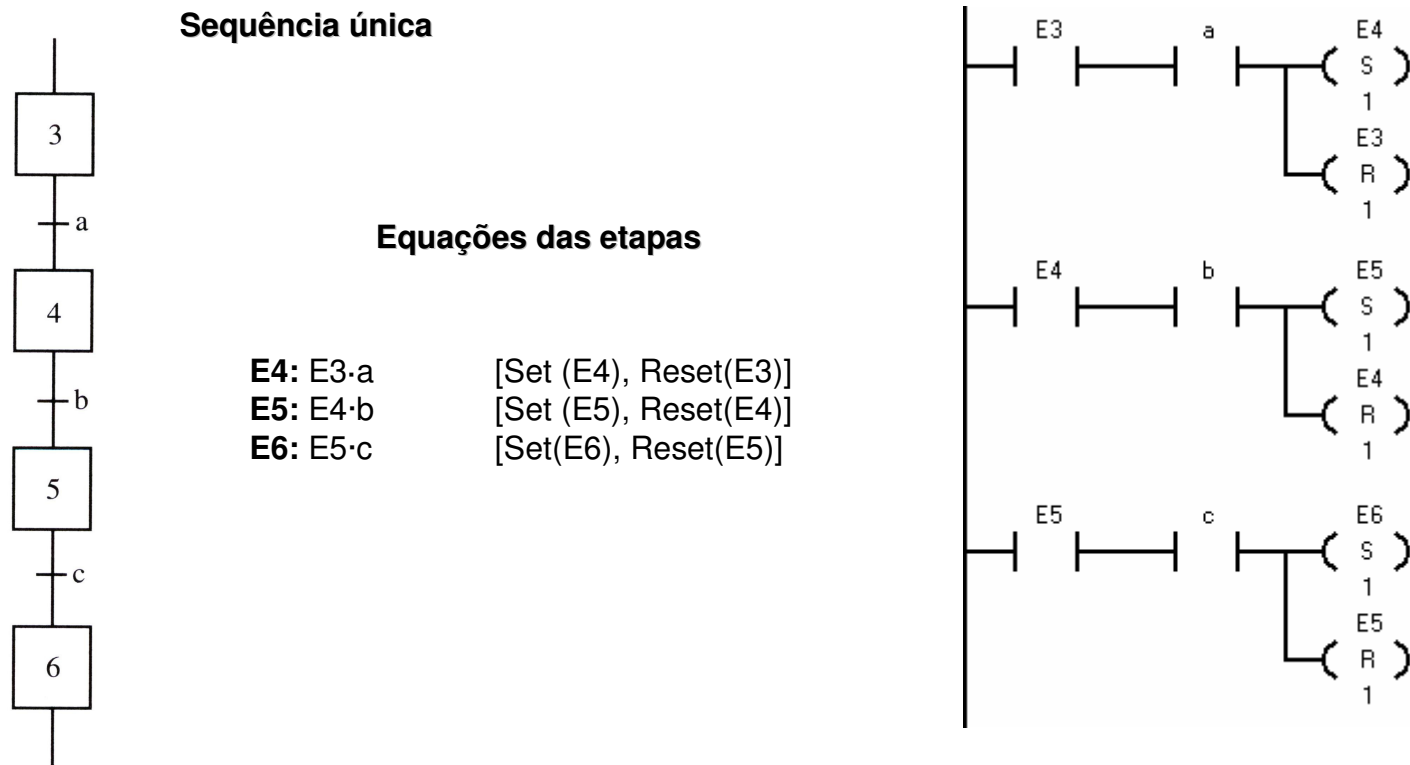
$$E_n: E_{n-1} \cdot t_{n-1} \quad [\text{Set}(E_n), \text{Reset}(E_{n-1})]$$

Com a etapa anterior activa e a respectiva transição verdadeira ($E_{n-1} \cdot t_{n-1} = 1$) pode-se activar a etapa seguinte, $E_n = 1$, e desactivar a etapa anterior, $E_{n-1} = 0$.
Utiliza-se a instrução SET para activar e a instrução RESET para desactivar etapas

Lógica programável

■ Programação em GRAFCET

➤ Metodologia I para programar o grafcet

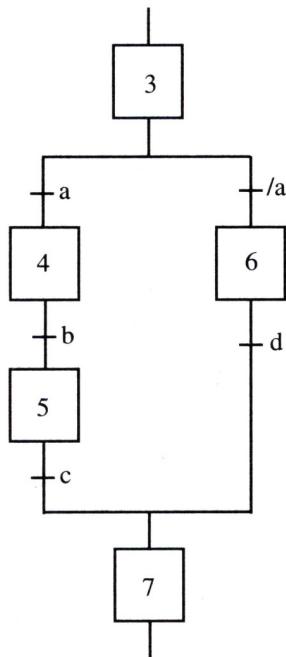


Lógica programável

■ Programação em GRAFCET

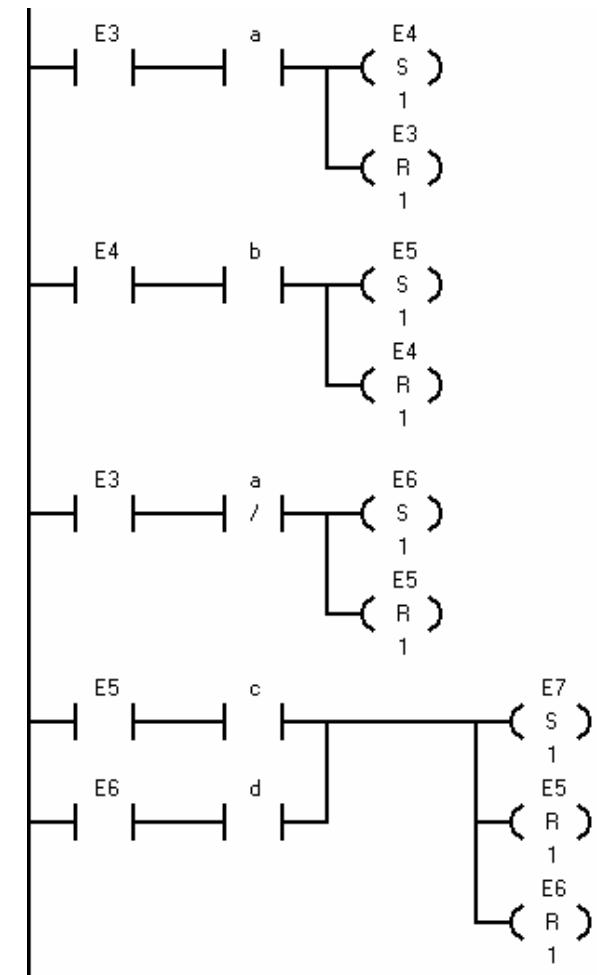
➤ Metodologia I para programar o grafcet

Sequências alternativas



Equações das etapas

E4: $E3 \cdot a$ [Set (E4), Reset(E3)]
E5: $E4 \cdot b$ [Set (E5), Reset(E4)]
E6: $E3 \cdot /a$ [Set(E6), Reset(E3)]
E7: $E5 \cdot c + E6 \cdot d$ [Set (E7), Reset(E5,E6)]

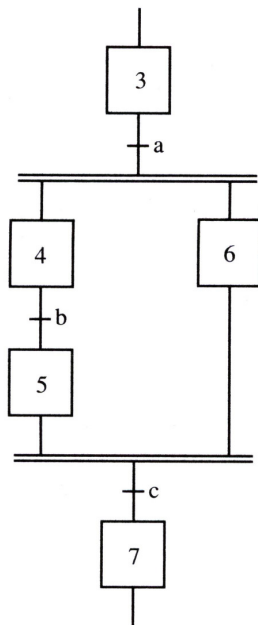


Lógica programável

■ Programação em GRAFCET

➤ Metodologia I para programar o grafcet

Sequências simultâneas

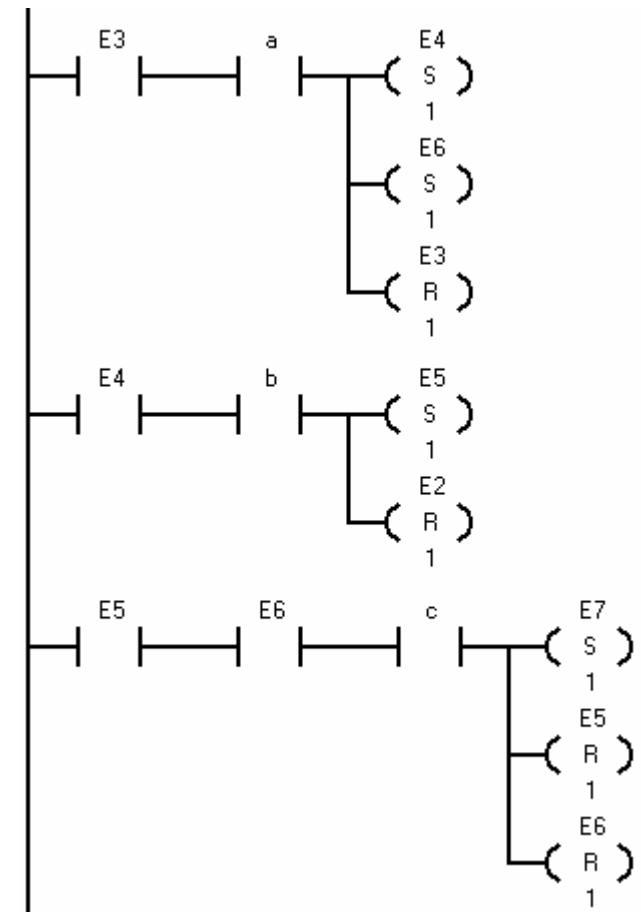


Equações das etapas

E4,E6: $E3 \cdot a$ [Set (E4,E6), Reset(E3)]

E5: $E4 \cdot b$ [Set (E5), Reset(E4)]

E7: $E5 \cdot E6 \cdot c$ [Set (E7), Reset(E5,E6)]

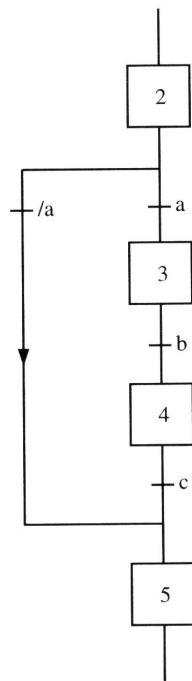


Lógica programável

■ Programação em GRAFCET

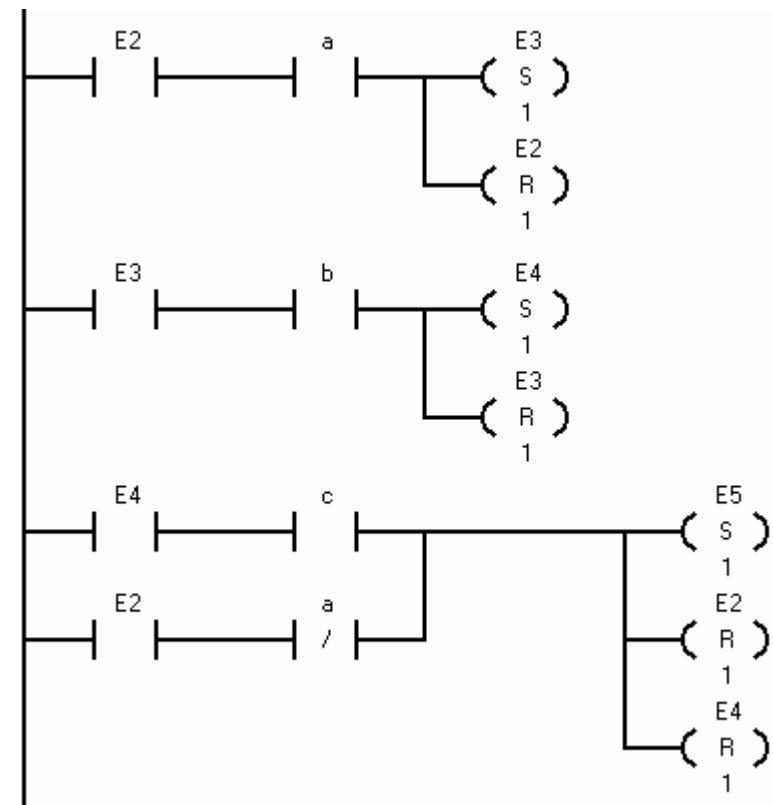
➤ Metodologia I para programar o grafcet

Salto de etapas



Equações das etapas

E3: $E2 \cdot a$ [Set (E3), Reset(E2)]
E4: $E3 \cdot b$ [Set (E4), Reset(E3)]
E5: $E2 \cdot /a + E4 \cdot c$ [Set (E5), Reset(E2, E4)]

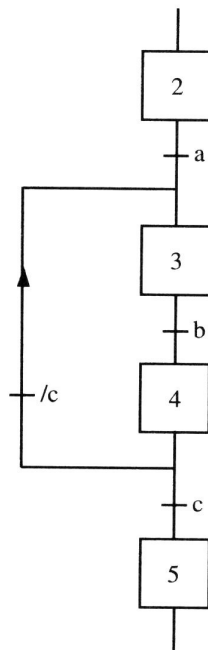


Lógica programável

■ Programação em GRAFCET

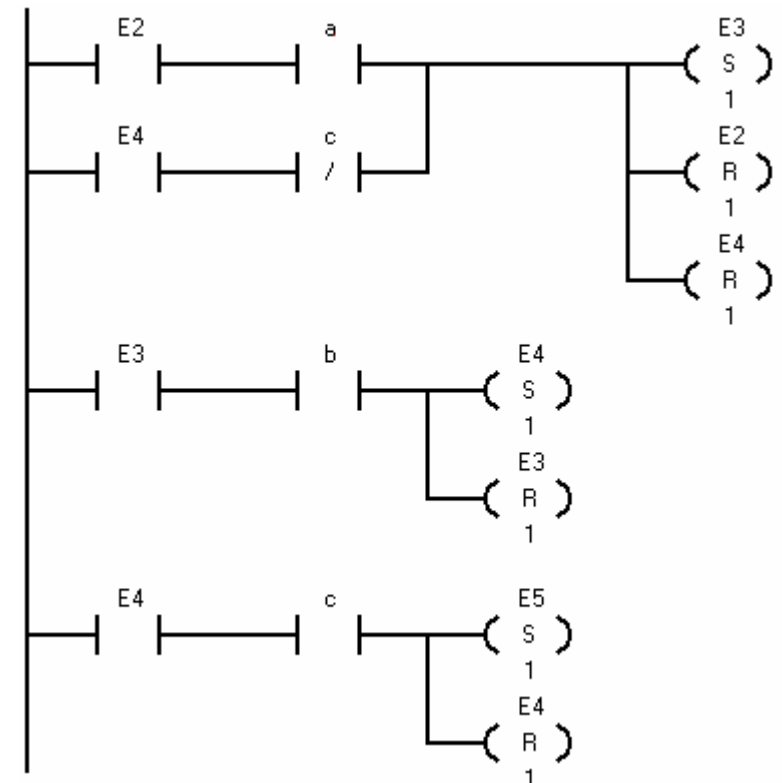
➤ Metodologia I para programar o grafcet

Repetição de etapas



Equações das etapas

E3: $E2 \cdot a + E4 \cdot /c$ [Set (E3), Reset(E2,E4)]
E4: $E3 \cdot b$ [Set (E4), Reset(E3)]
E5: $E4 \cdot c$ [Set (E5), Reset(E4)]

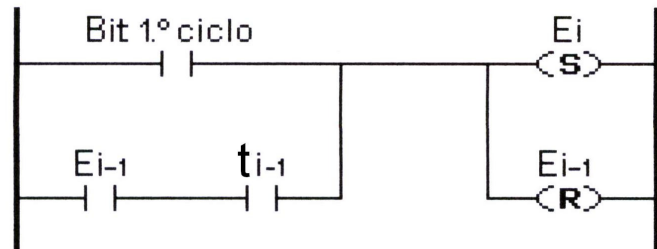


Lógica programável

■ Programação em GRAFCET

➤ Metodologia I para programar o grafcet

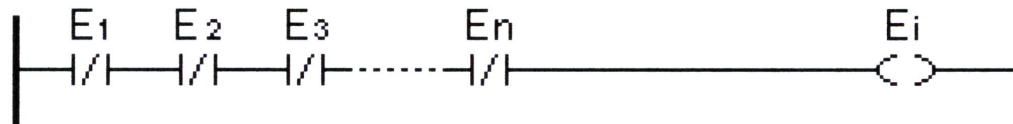
Etapa Inicial (S7-200)



Ei: Bit 1.º ciclo + $E_{i-1} \bullet t_{i-1}$ [Set (E_i), Reset (E_{i-1})]

A etapa inicial, E_i , é activada pelo bit do 1º ciclo de scan, que nos autómatos s7-200 é o SM0.1, ou pelo retorno da última etapa do grafcet, E_{i-1} .

Etapa Inicial (alternativa)



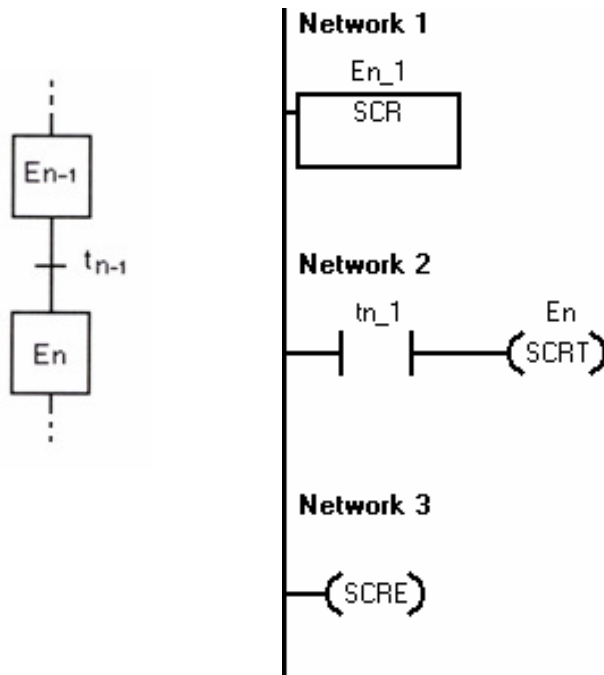
Quando os autómatos não possuem bit de 1º ciclo, a etapa inicial, E_i , pode ser activada quando todas as outras etapas estiverem inactivas.

Lógica programável

■ Programação em GRAFCET

➤ Metodologia II para programar grafcet (Relés de Controlo Sequencial)

Caso Genérico



São utilizadas 3 instruções distintas de relés de controlo sequencial, onde cada “relé” corresponde a uma etapa, nomeadamente:

SCR – abre a etapa

SCRT – transita de etapa. Esta instrução desactiva a etapa actual e activa a próxima

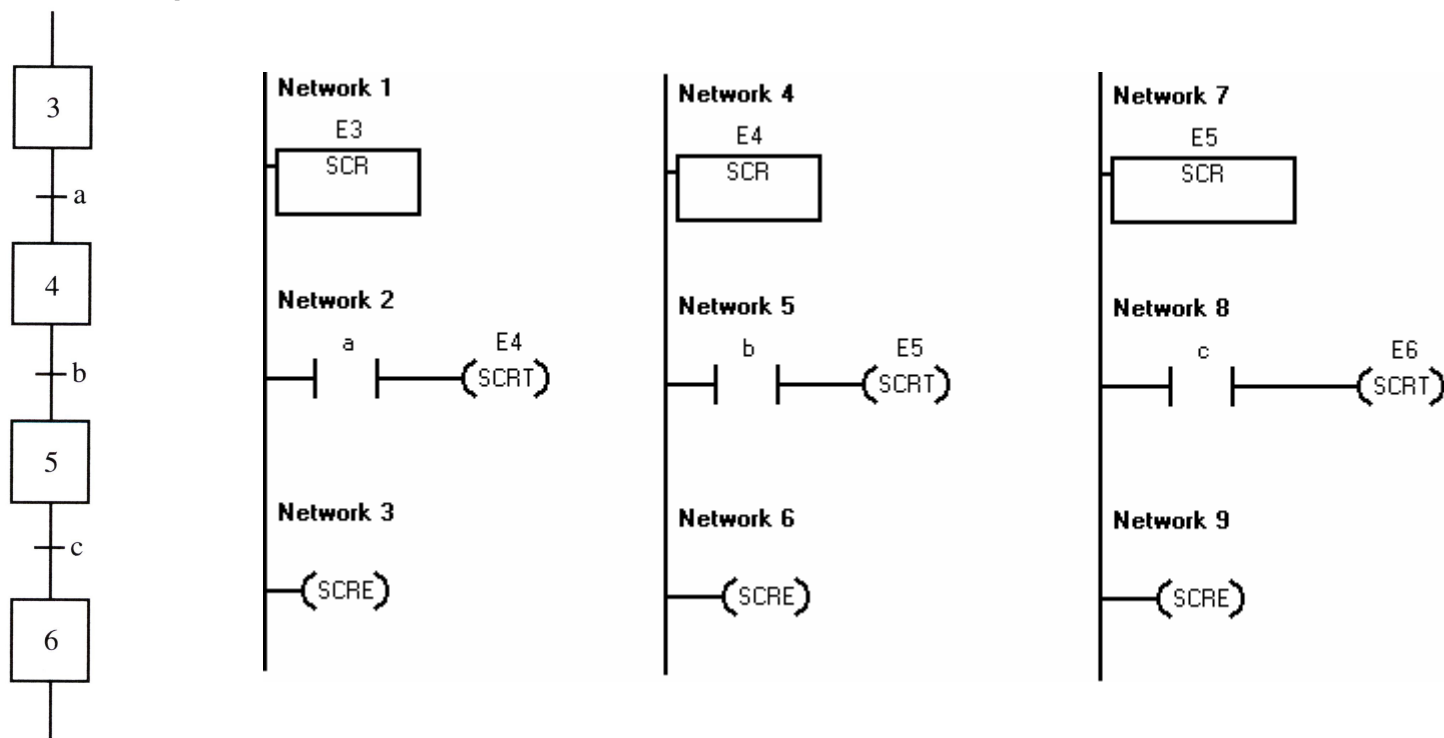
SCRE – fecha a etapa

Lógica programável

- Programação em GRAFCET

- Metodologia II para programar grafcet (Relés de Controlo Sequencial)

Sequência única

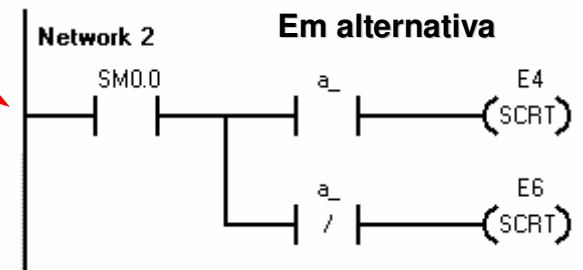
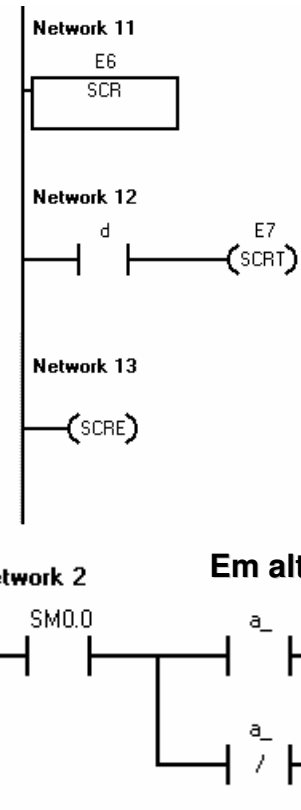
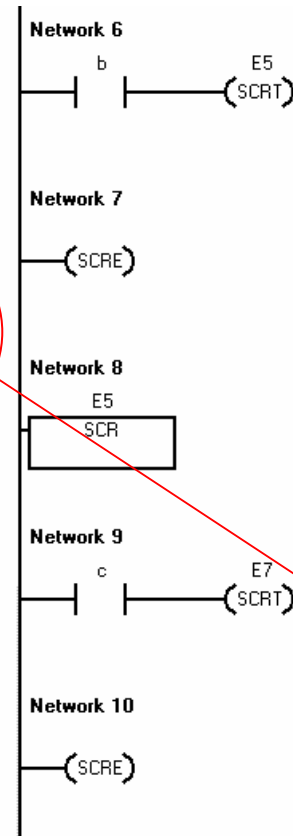
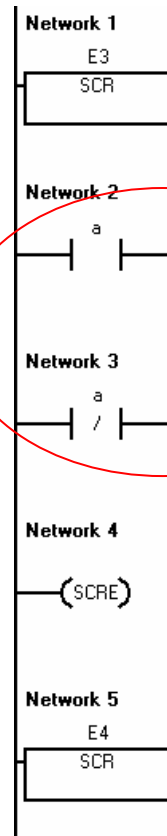
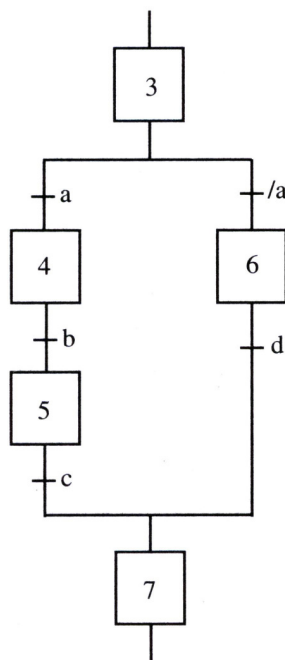


Lógica programável

■ Programação em GRAFCET

➤ Metodologia II para programar grafcet (Relés de Controlo Sequencial)

Sequências alternativas

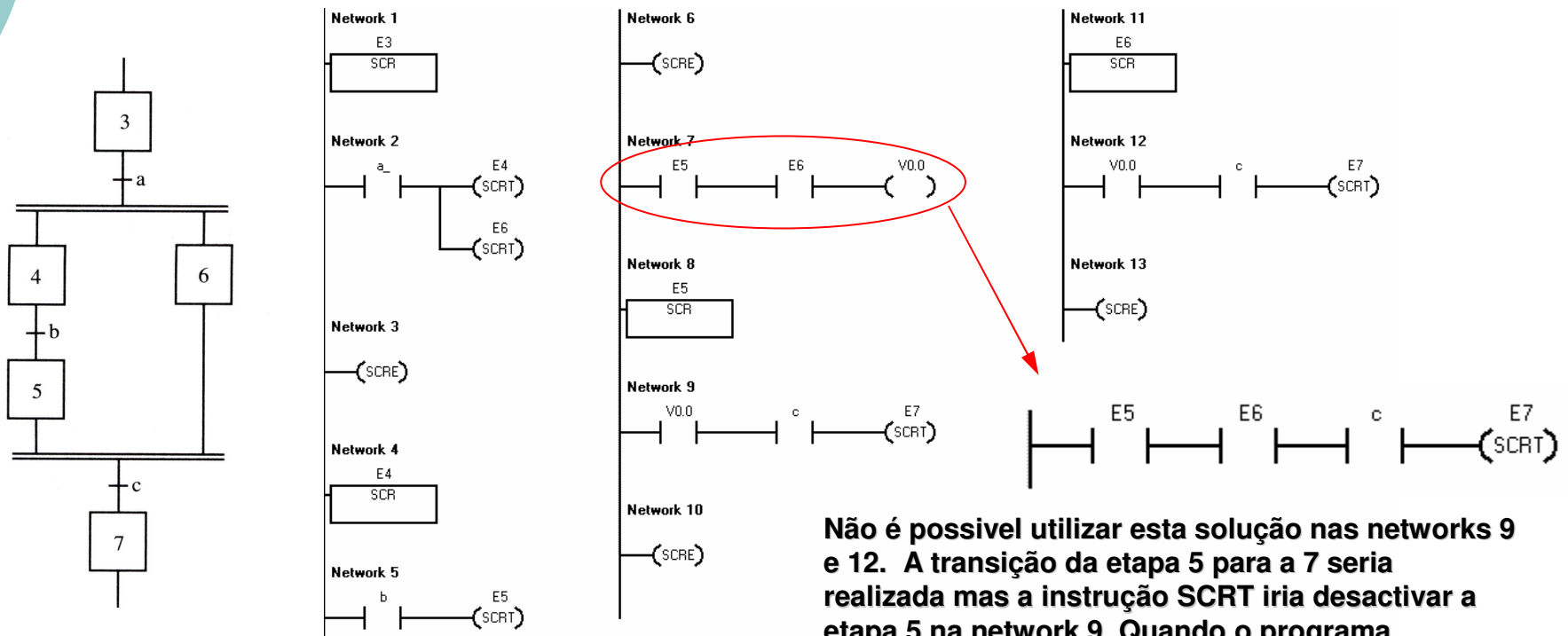


Lógica programável

■ Programação em GRAFCET

➤ Metodologia II para programar grafcet (Relés de Controlo Sequencial)

Sequências simultâneas

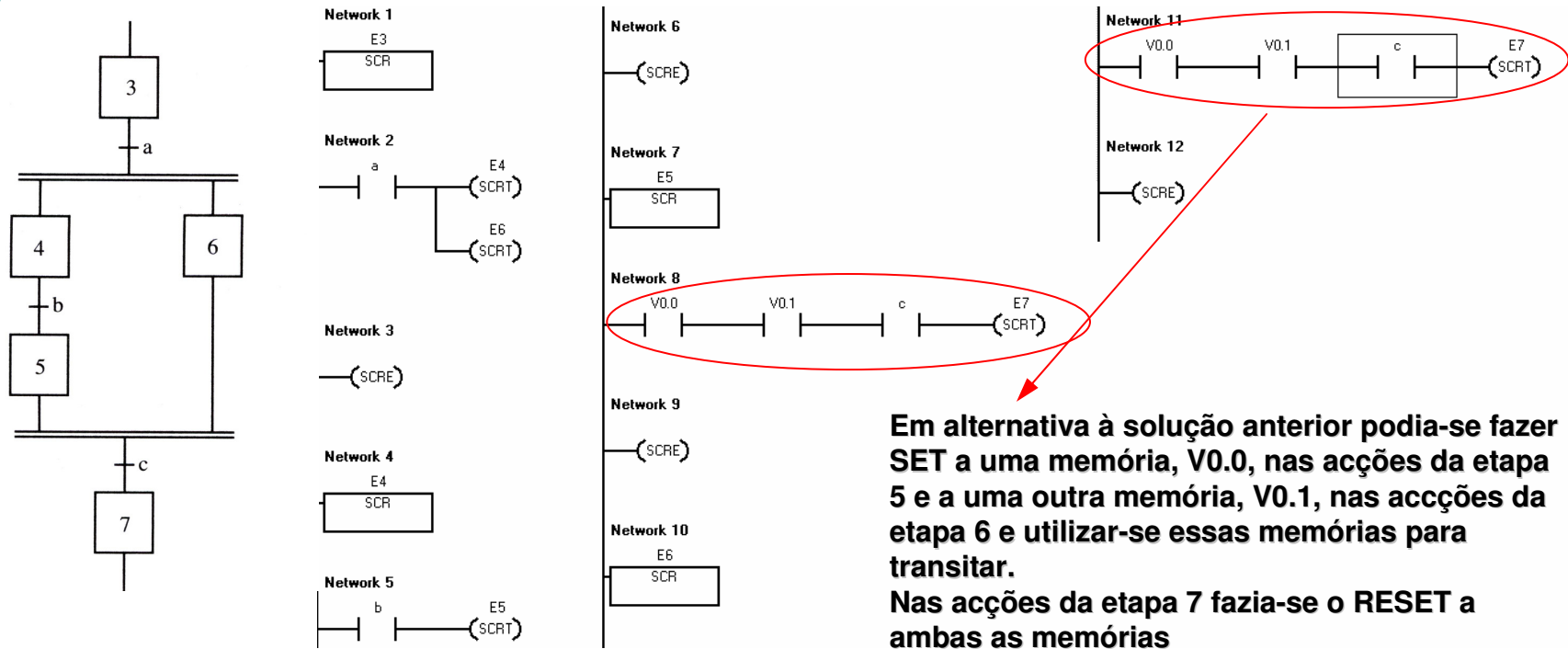


Lógica programável

■ Programação em GRAFCET

➤ Metodologia II para programar grafcet (Relés de Controlo Sequencial)

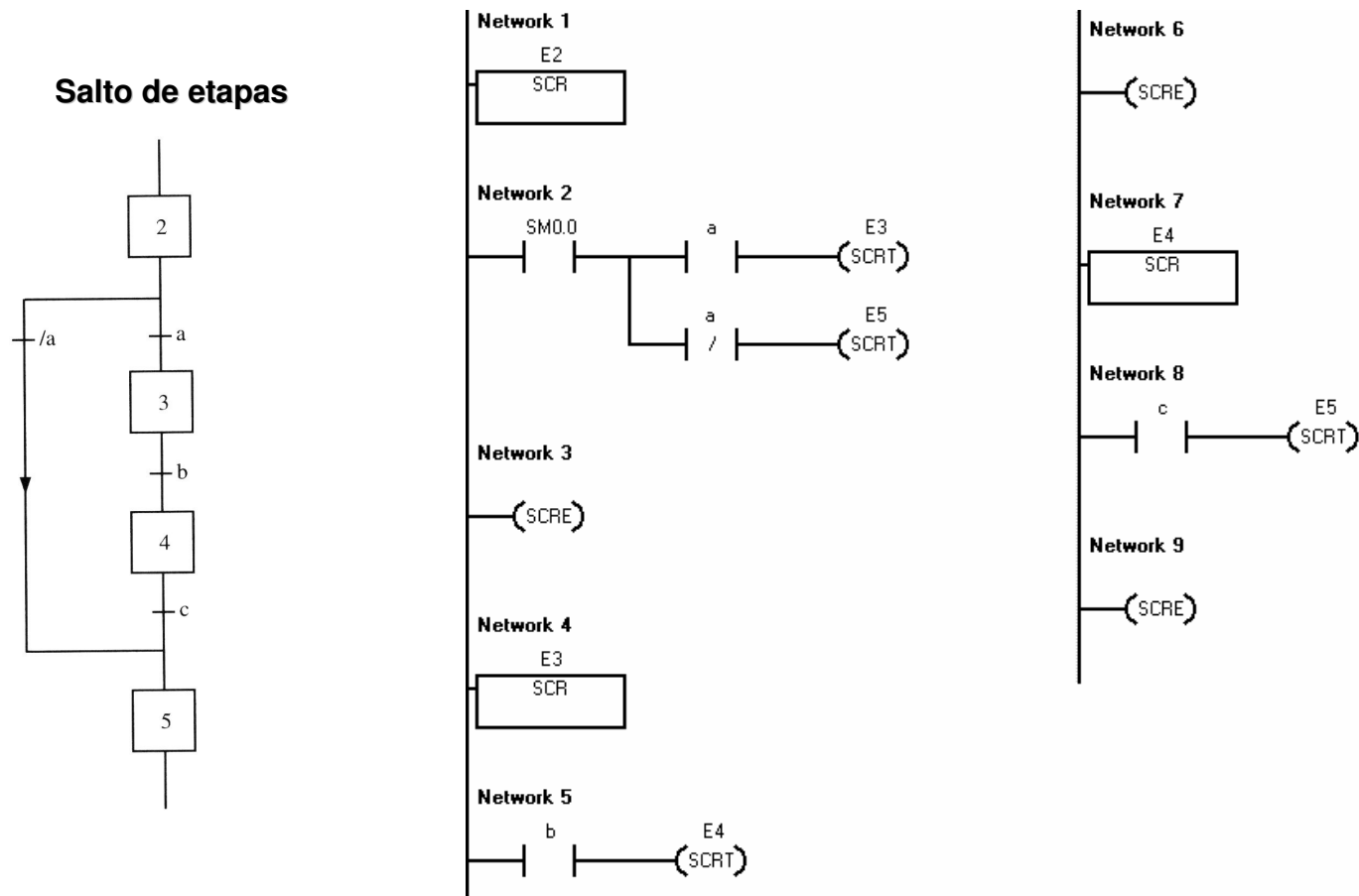
Sequências simultâneas



Lógica programável

■ Programação em GRAFCET

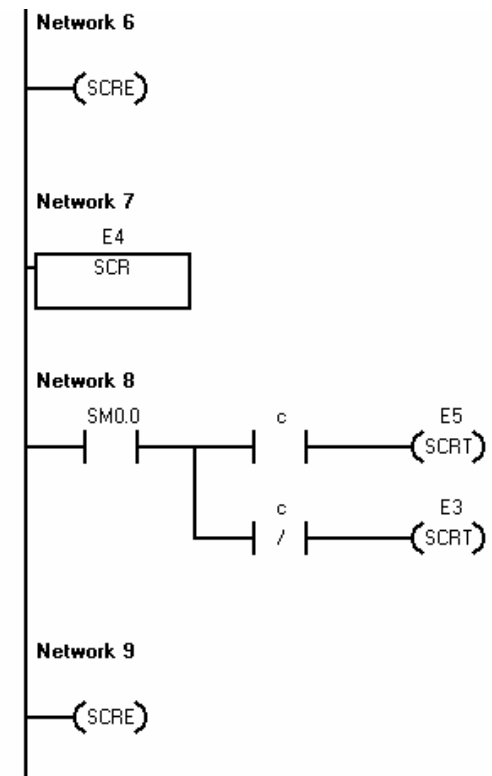
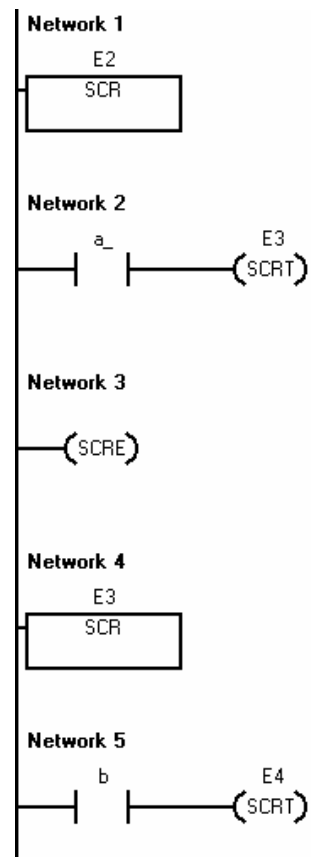
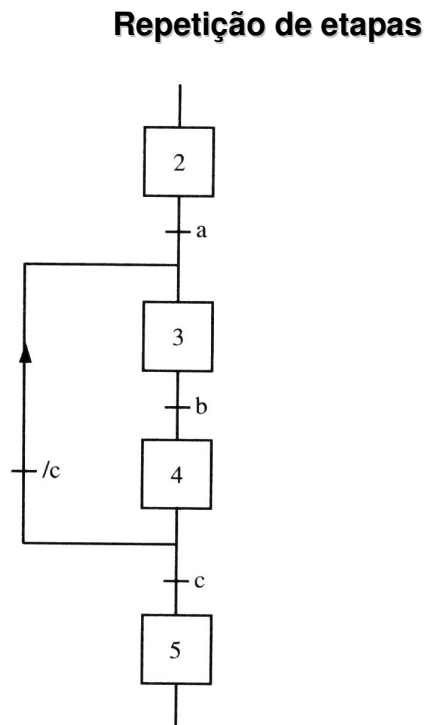
➤ Metodologia II para programar grafcet (Relés de Controlo Sequencial)



Lógica programável

■ Programação em GRAFCET

➤ Metodologia II para programar grafcet (Relés de Controlo Sequencial)

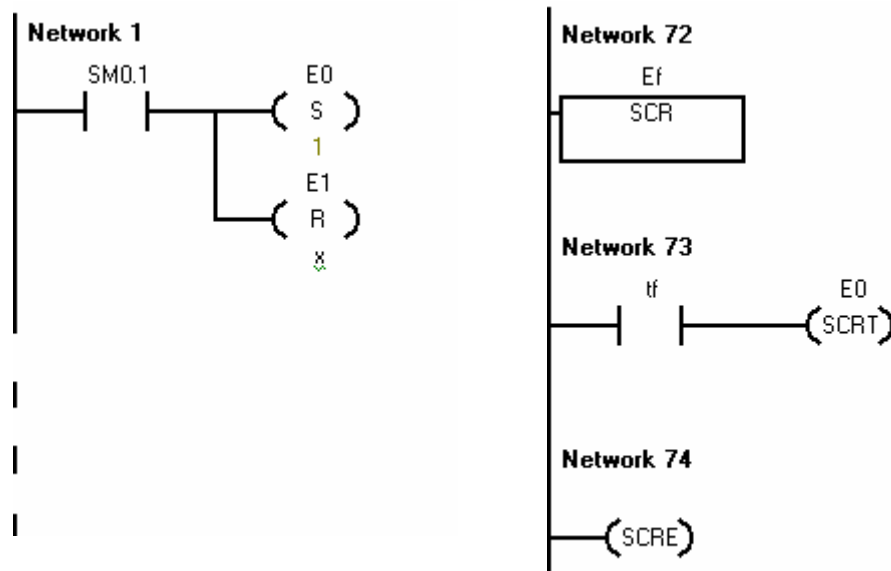


Lógica programável

■ Programação em GRAFCET

➤ Metodologia II para programar grafcet (Relés de Controlo Sequencial)

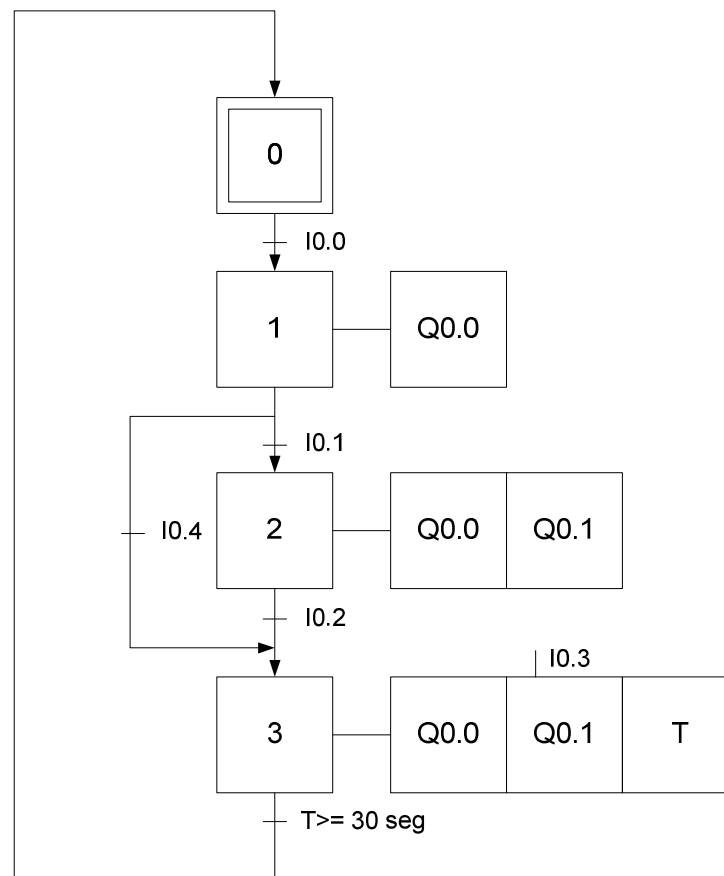
Etapa Inicial (S7-200)



A etapa inicial, E0, é activada pelo bit do 1º ciclo de scan, que nos autómatos s7-200 é o SM0.1, ou pelo retorno da última etapa do grafcet, Ef.

Lógica programável

- Programação em GRAFCET



EXERCICIO

Dado o grafcet, implemente a sua programação no automático, por ambos os métodos vistos



Lógica programável

- **Programação em GRAFCET**

- **Implementação de paragem de emergência num grafcet**

Existem diferentes maneiras de se implementar a paragem de emergência num grafcet, mediante a necessidade do processo.

Em determinadas situações será possível obter soluções mais simplificadas optando por utilizar grafkets diferentes e ligados para o funcionamento normal e para a paragem de emergência, noutras será mais simples implementar o funcionamento e a paragem num único grafcet.

Para perceber melhor este conceito estuda-se de seguida um exemplo de um processo, onde variam as necessidades da paragem de emergência, consoante o produto ao qual é aplicado, e onde se apresentam diversas soluções com diferentes graus de complexidade.

Lógica programável

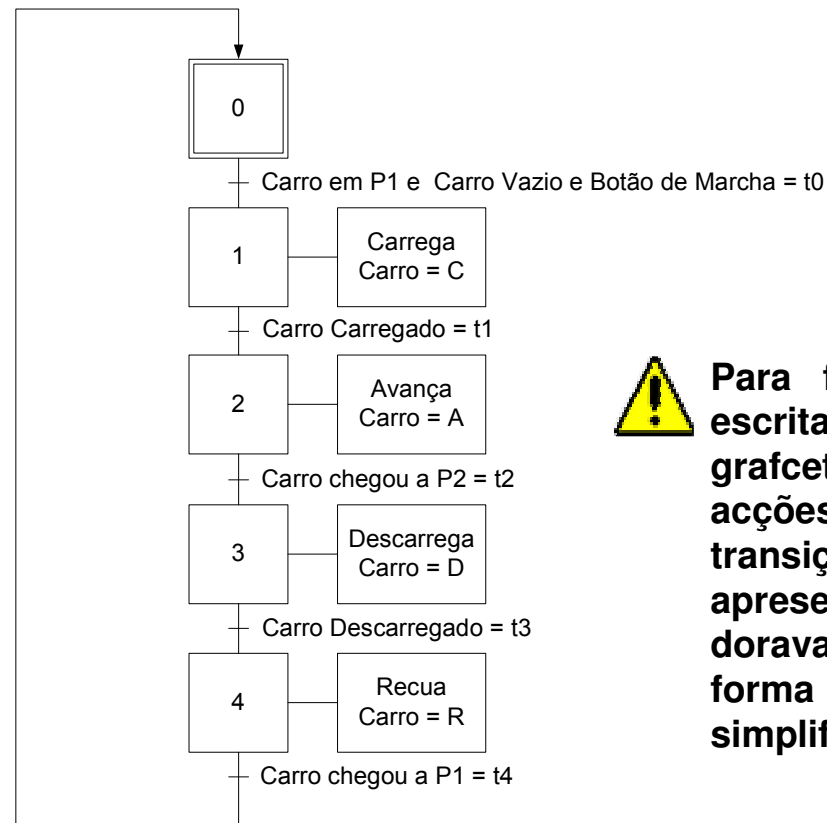
■ Programação em GRAFCET

➤ Paragem de Emergência- Exemplo

Considere um processo no qual existem 2 postos de trabalho, P1 e P2, distantes um do outro, e um veículo que transporta material desde P1, onde é carregado, até P2, onde é descarregado e que depois retorna a P1.

Quando a paragem de emergência (PE) é **activada** cessam todos os movimentos e é activada a sinalização luminosa.

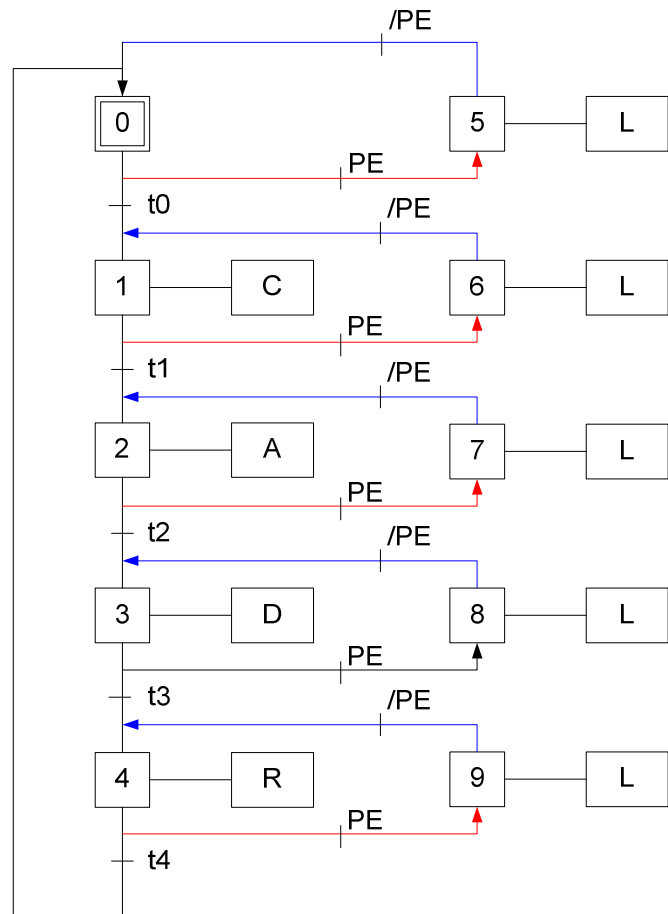
Quando é **desactivada** podem ocorrer **4 situações** distintas.



Para facilitar a escrita do grafcet, as acções e as transições apresentam-se doravante de forma simplificada.

Lógica programável

■ Programação em GRAFCET



➤ 1ª situação:

Carro transporta parafusos, que não se deterioram ao longo do tempo, logo continua a fazer o que estava antes da PE.

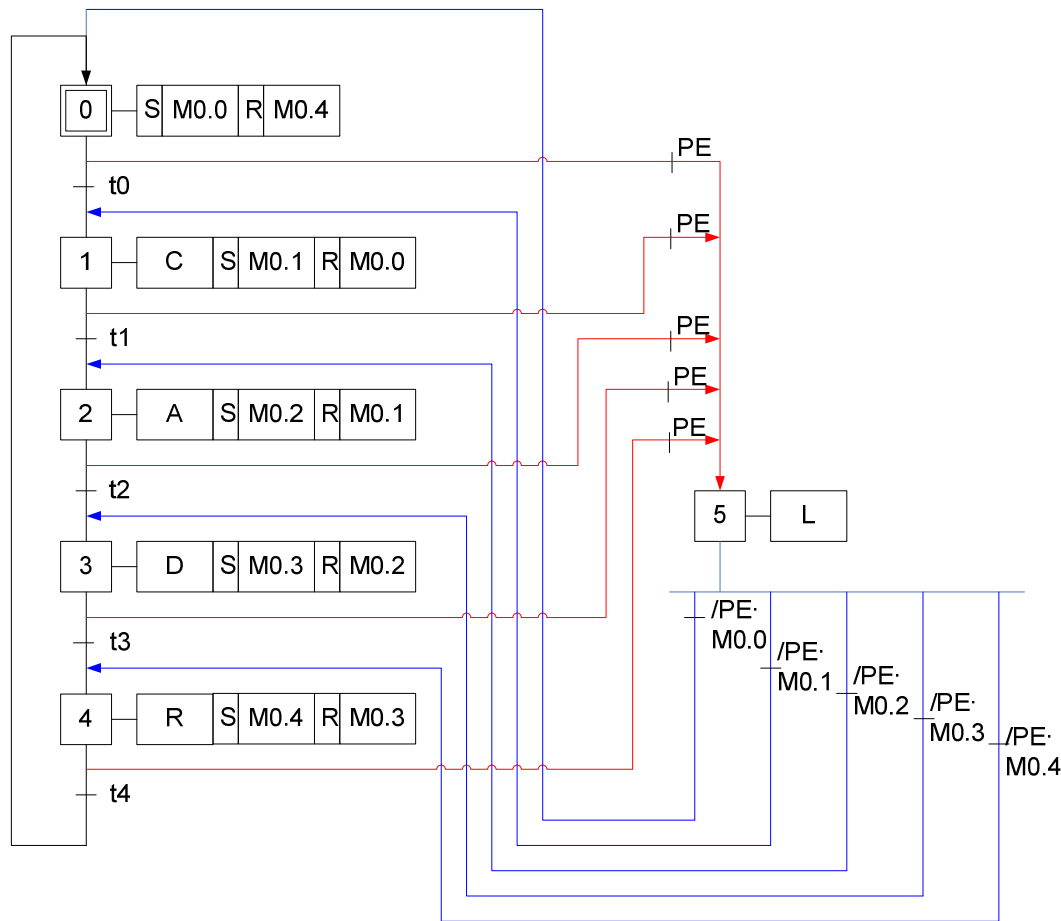
➤ 1ª solução.

Em todas as etapas, sempre que aparece a PE transita-se para outra etapa, equivalente, onde não se realiza a acção e se sinaliza a PE.

É uma solução confusa e trabalhosa.

Lógica programável

■ Programação em GRAFCET



➤ 1ª situação:

Carro transporta parafusos, que não se deterioram ao longo do tempo, logo continua a fazer o que estava antes da P.E.

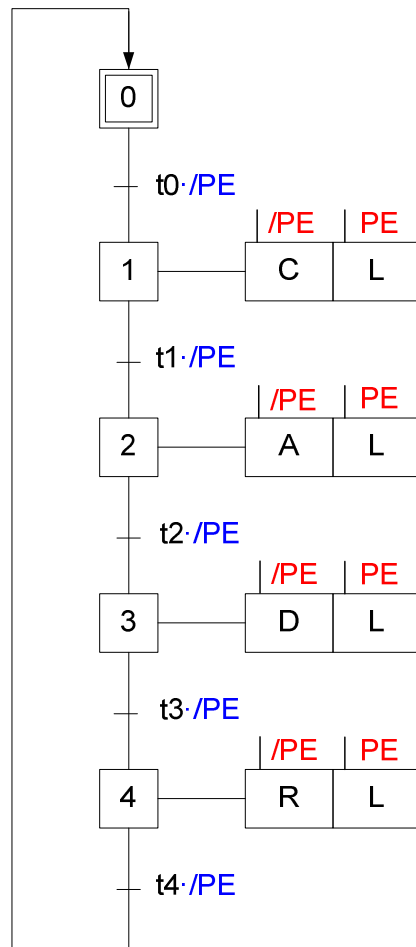
➤ 2ª solução.

Em vez de repetir todas as etapas, cria-se uma nova, onde se sinaliza e não se realiza nenhuma das outras acções.

É uma solução muito confusa.

Lógica programável

■ Programação em GRAFCET



➤ 1ª situação:

Carro transporta parafusos, que não se deterioram ao longo do tempo, logo continua a fazer o que estava antes da P.E.

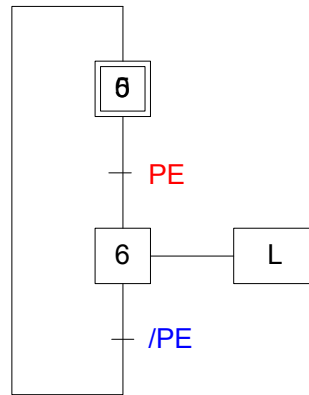
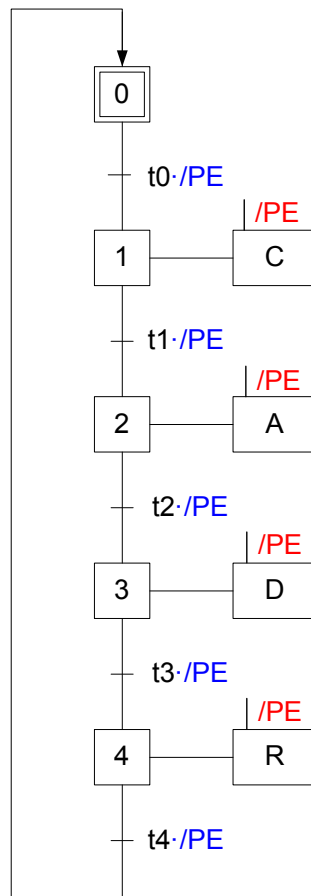
➤ 3ª solução.

A maneira mais simples de um grafcet retomar o estava a fazer antes da PE é não deixar que o grafcet transite de etapa, colocando a condição /PE na transição. Para além disso como é necessário cessar a acção enquanto houver PE, condiciona-se as saídas e acrescenta-se a sinalização luminosa também condicionado ao facto de haver PE.

É uma solução mais simples que as anteriores, mas mesmo assim algo repetitiva.

Lógica programável

■ Programação em GRAFCET



➤ 1ª situação:

Carro transporta parafusos, que não se deterioram ao longo do tempo, logo continua a fazer o que estava antes da PE.

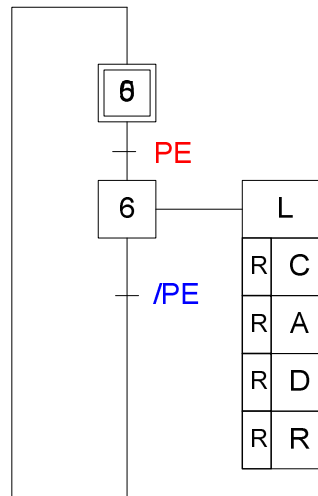
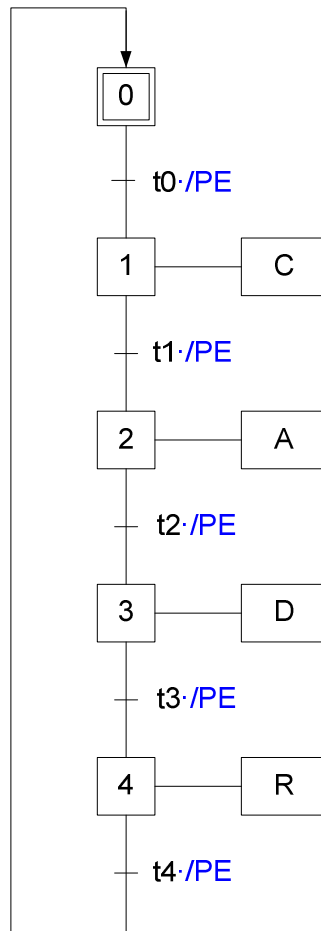
➤ 4ª solução.

Cria-se um grafcet para a PE, que funciona em simultâneo com o grafcet principal. No grafcet principal continua-se a não deixar mudar de etapa, caso exista PE.

É uma solução simples, mas ainda assim algo repetitivo, pois apresenta todas as saídas condicionadas.

Lógica programável

■ Programação em GRAFCET



➤ 1ª situação:

Carro transporta parafusos, que não se deterioram ao longo do tempo, logo continua a fazer o que estava antes da PE.

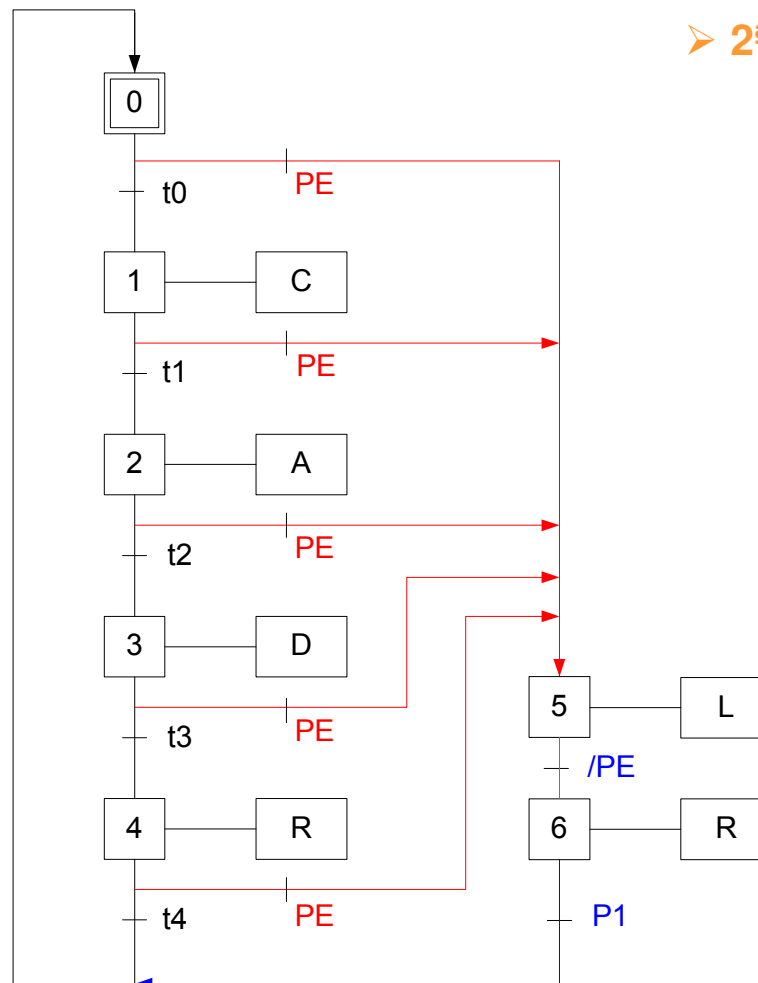
➤ 5ª solução.

Faz-se o reset a todas as acções no grafcet da PE.

É a solução mais simples e menos repetitiva, mas é necessário ter o cuidado de programar as instruções de reset depois das activações das saídas.

Lógica programável

■ Programação em GRAFCET



➤ 2ª situação:

Carro transporta gelado, que se deteriora, logo quando a PE deixar de existir, retorna a P1 para ser descarregado manualmente.

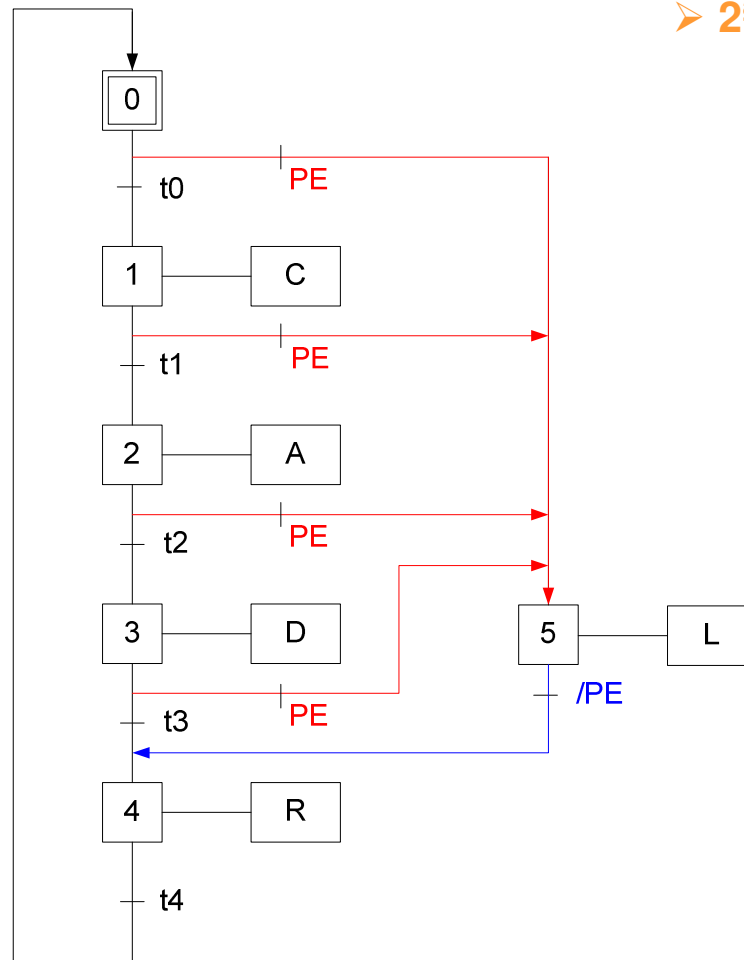
➤ 1ª solução.

Sempre que existir PE criam-se ramos alternativos que obrigam a transitar para a etapa 5 onde se sinaliza a PE. Quando esta deixar de existir retorna-se ao posto 1.

É uma solução trabalhosa. Se o grafcet tiver muitas etapas fica confuso.

Lógica programável

■ Programação em GRAFCET



➤ 2ª situação:

Carro transporta gelado, que se deteriora, logo quando a PE deixar de existir, retorna a P1 para ser descarregado manualmente.

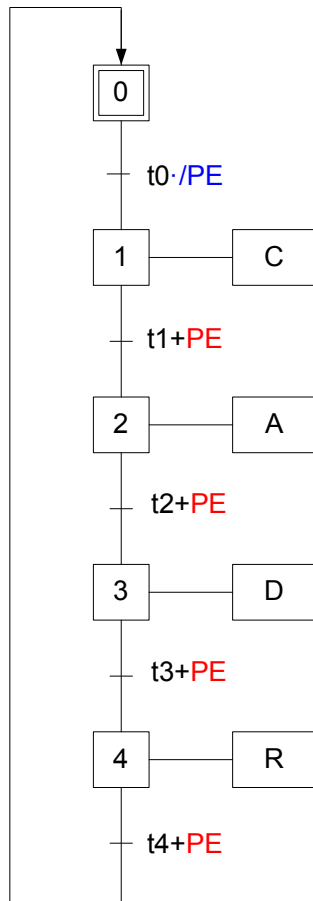
➤ 2ª solução.

Como existe uma etapa onde se obriga o veículo a recuar até ao posto 1 (4) pode-se levar o grafcet para esta etapa quando a PE deixar de existir.

Continua a ser uma solução trabalhosa e continua a ficar confuso se o grafcet tiver muitas etapas.

Lógica programável

■ Programação em GRAFCET



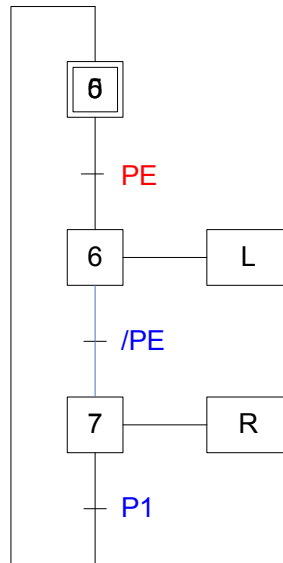
➤ 2ª situação:

Carro transporta gelado, que se deteriora, logo quando a PE deixar de existir, retorna a P1 para ser descarregado manualmente.

➤ 3ª solução.

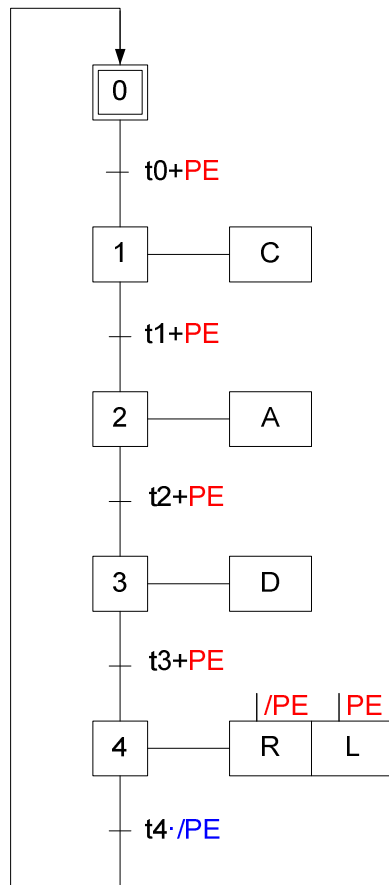
Cria-se um grafcet para a PE, que funciona em simultâneo com o grafcet principal. Como o grafcet da PE se encarrega de fazer recuar o veículo até P1, o grafcet principal tem de ser colocado no repouso e

não pode sair de lá enquanto existir PE. Para tal basta adicionar a condição “ou PE” em todas as transições, o que obriga o grafcet a transitar até chegar à etapa 0 e a condição “e /PE” à transição t0. Como as acções devem ser programadas depois de todas as transições, não se realiza nenhuma, pois nenhum das etapas respectivas está activa.



Lógica programável

■ Programação em GRAFCET



➤ 2ª situação:

Carro transporta gelado, que se deteriora, logo quando a PE deixar de existir, retorna a P1 para ser descarregado manualmente.

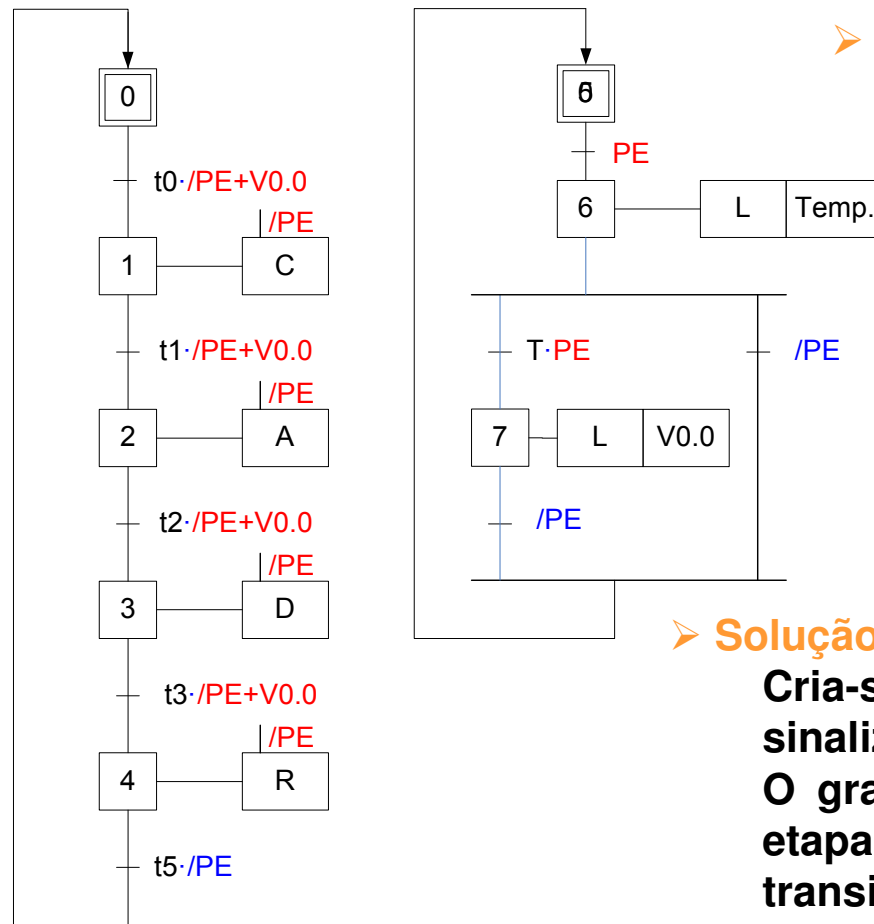
➤ 4ª solução.

Utilizando a técnica de forçar o grafcet a transitar de etapa com a adição da condição “ou PE” é possível evoluir até à etapa 4 onde se sinaliza se houver PE e se deixa de fazer a acção recuar enquanto houver PE.

É a solução mais simples.

Lógica programável

■ Programação em GRAFCET



➤ 3ª situação:

Carro transporta manteiga, que se deteriora ao longo do tempo, mas não imediatamente, logo quando PE deixar de existir, verifica qual o tempo decorrido. Se for superior a 1 minuto, o carro retorna a P1 para ser descarregado manualmente, mas se for inferior continua com a acção que estava a fazer.

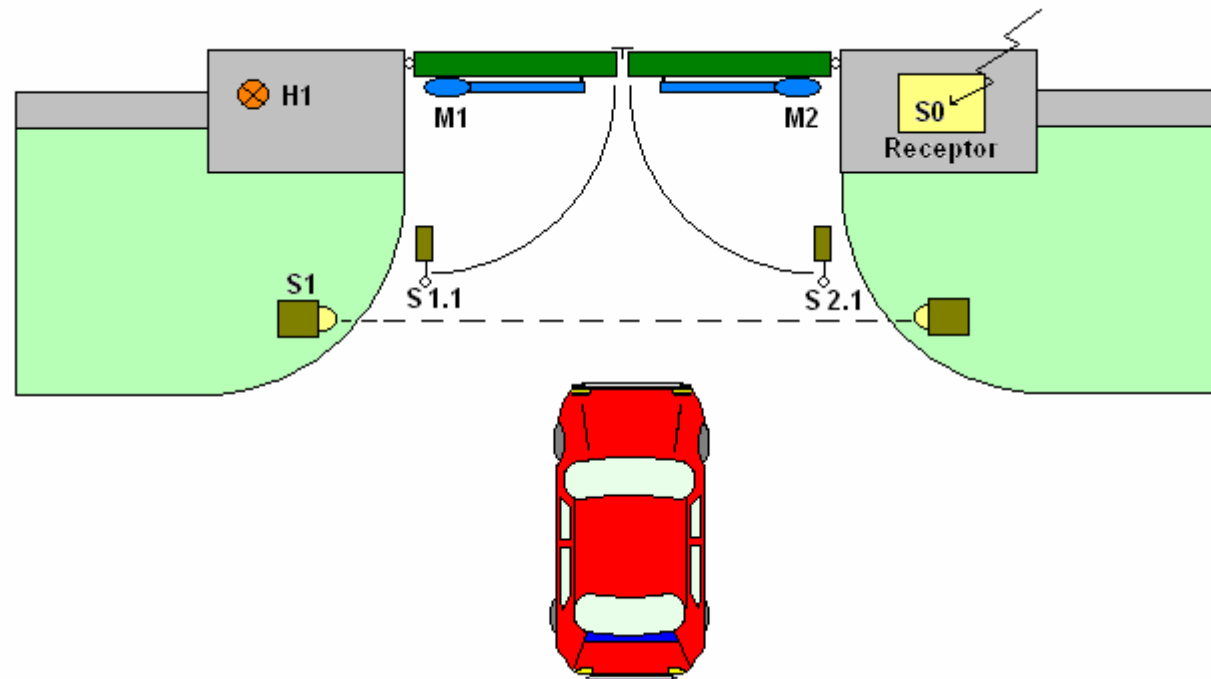
➤ Solução.

Cria-se um grafcet para PE onde se sinaliza e se verifica o tempo decorrido. O grafcet principal permanece na mesma etapa enquanto não passar 1 minuto ou transita para a etapa 4 onde irá recuar, quando deixar de existir PE.

Lógica programável

- Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET

- Exercício de Aplicação N°3 - Portões Automáticos





Lógica programável

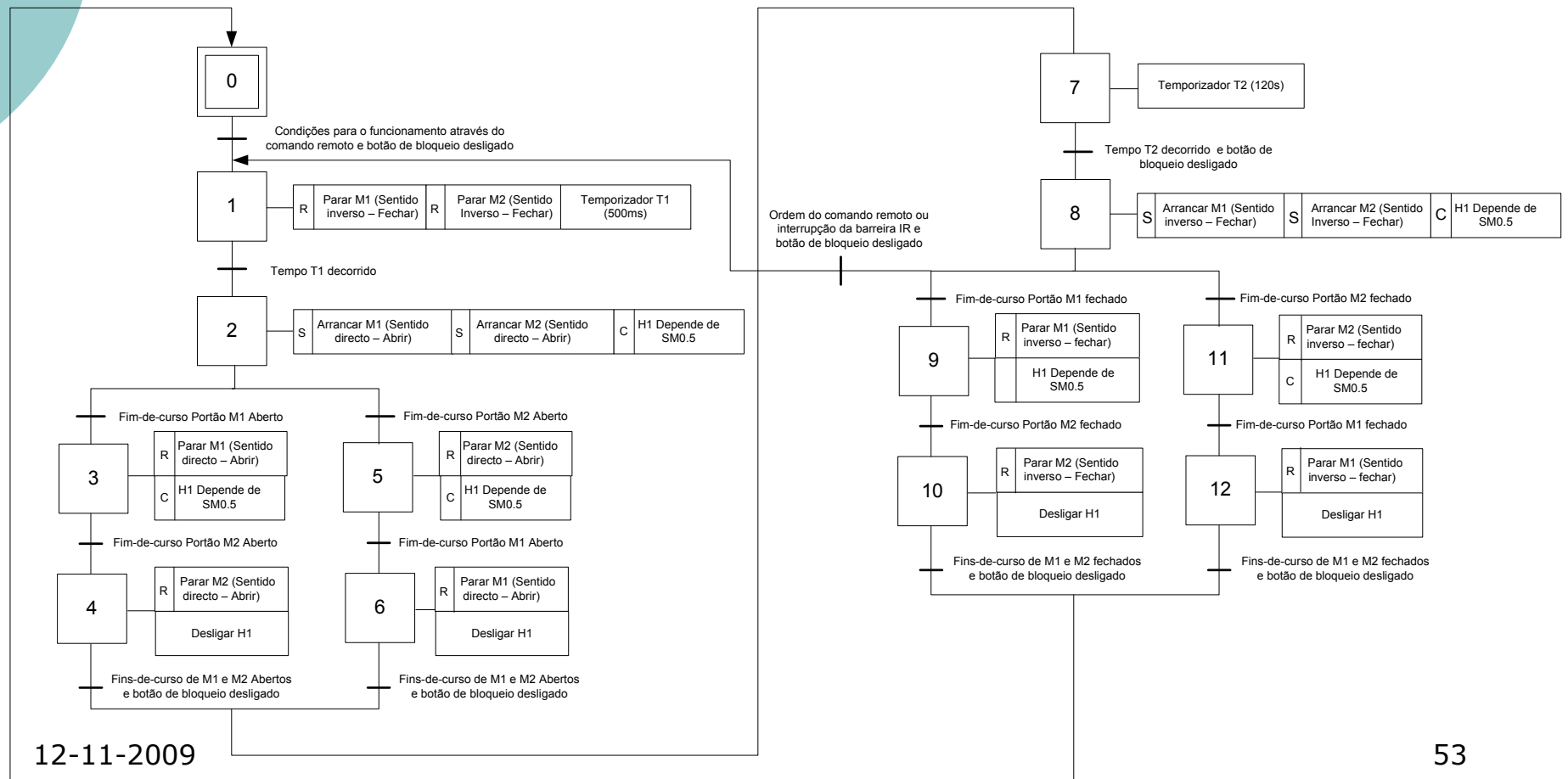
- **Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET**

- **Exercício de Aplicação Nº 3 – Descrição**

- A ordem de abertura dos portões é dada por um emissor de comando à distância. Quando o sinal é recebido pelo receptor actua o contacto S0 e abre o portão actuando directamente sobre M1 e M2 (KM1.1 e KM2.1).
- O sinalizador H1 acende de forma intermitente sempre que os portões abrem e fecham. Considere a existência de um botão que evita a abertura e fecho dos portões (S3).
- O final da abertura dos portões é dado pela actuação dos fins de curso S1.1 e S2.1. O final do fecho dos portões é dado por dois interruptores de excesso de binário, cada um associado a um motor (S1.2 e S2.2).
- Os portões permanecem abertos durante 2 minutos ao fim do qual voltam a fechar. Se durante o fecho dos portões a barreira de infravermelhos for interrompida ele vai voltar a abrir. O fecho dos portões corresponde a actuar KM1.2 e KM2.2

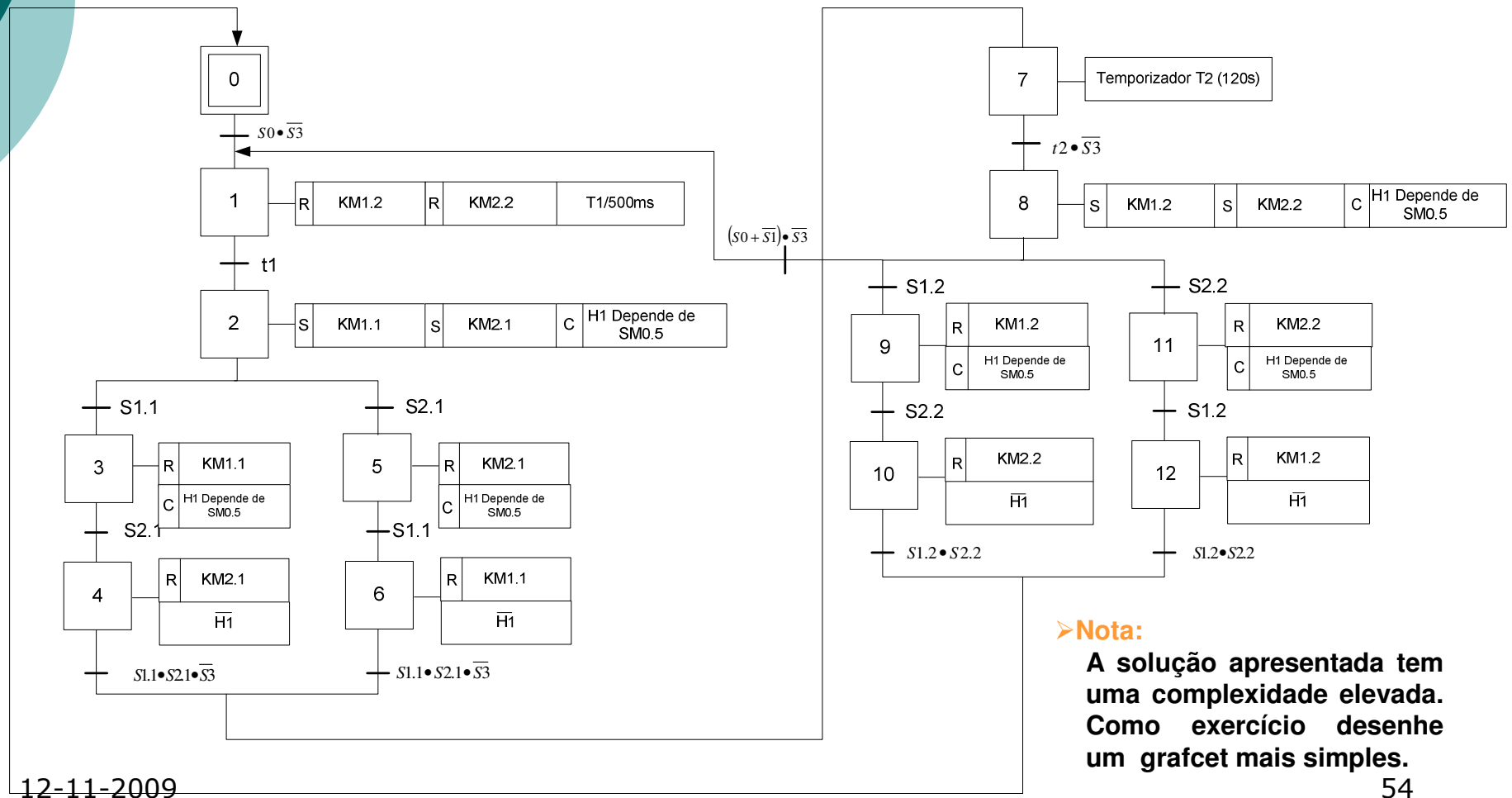
Lógica programável

■ Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET – Solução Gráfica – Nível 1



Lógica programável

■ Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET – Solução Gráfica – Nível 2



Lógica programável

- **Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET**

- **Exercício de Aplicação Nº 3 – Solução usando linguagem de contactos (I)**

Símbolos usados – Etapas e Transições

Etapas	Bits		Transições	Entradas	Descrição
E0	M0.0		S0	I0.0	Contacto do receptor do telecomando
E1	M0.1		S1	I0.1	Contacto da Barreira IR
E2	M0.2		S1.1	I0.2	Fim de curso Motor 1 (Portão aberto)
E3	M0.3		S1.2	I0.3	Fim de curso Motor 2 (Portão aberto)
E4	M0.4		S2.1	I0.4	Interruptor binário excessivo Motor 1 (Portão fechado)
E5	M0.5		S2.2	I0.5	Interruptor binário excessivo Motor 2 (Portão fechado)
E6	M0.6		S3	I0.6	Inibição do funcionamento dos portões
E7	M0.7		t1	Bit T37	Tempo de espera entre paragem e arranque
E8	M1.0		t2	Bit T38	Tempo que o portão permanece aberto
E9	M1.1				
E10	M1.2				
E11	M1.3				
E12	M1.4				

Lógica programável

- **Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET**

- **Exercício de Aplicação Nº 3 – Solução usando linguagem de contactos (I)**

- Símbolos usados - Acções**

Acções	Bits	Condição	Saídas	Descrição
Set KM1.1	M0.2	-	Q0.0	Ligar o contactor para abrir o portão (M1)
Set KM1.2	M1.0	-	Q0.1	Ligar o contactor para abrir o portão (M2)
Set KM2.1	M0.2	-	Q0.2	Ligar o contactor para fechar o portão (M1)
Set KM2.2	M1.0	-	Q0.3	Ligar o contactor para fechar o portão (M2)
Reset KM1.1	M0.3; M0.6	-	Q0.0	Desligar o contactor para abrir o portão (M1)
Reset KM1.2	M0.1; M1.1; M1.4	-	Q0.1	Desligar o contactor para abrir o portão (M2)
Reset KM2.1	M0.4; M0.5	-	Q0.2	Desligar o contactor para fechar o portão (M1)
Reset KM2.2	M0.1; M1.2; M1.3	-	Q0.3	Desligar o contactor para fechar o portão (M2)
H1	M0.2; M0.3; M0.5; M1.0; M1.1; M1.3	SM0.5	Q0.4	Sinalizador H1 intermitente
T1	M0.1	-	-	Tempo de espera entre paragem e arranque
T2	M0.7	-	-	Temporizador de abertura de portão

Lógica programável

- **Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET**

- **Exercício de Aplicação Nº 3 – Solução usando linguagem de contactos (I)**

Equações das etapas - Símbolos

E0	Bit 1º ciclo + ((E10+E12)·S1.2·S2.2)	Set E0; Reset E10; Reset E12
E1	$E0 \cdot S0 \cdot \overline{S3} + E8 \cdot (S0 + \overline{S1}) \cdot \overline{S3}$	Set E1; Reset E0; Reset E8
E2	$E1 \cdot t1$	Set E2; Reset E1
E3	$E2 \cdot S1.1$	Set E3; Reset E2
E4	$E3 \cdot S2.1$	Set E4; Reset E3
E5	$E2 \cdot S2.1$	Set E5; Reset E2
E6	$E5 \cdot S1.1$	Set E6; Reset E5
E7	$(E4 + E6) \cdot S1.1 \cdot S2.1 \cdot \overline{S3}$	Set E7; Reset E4; Reset E6
E8	$E7 \cdot t2 \cdot \overline{S3}$	Set E8; Reset E7
E9	$E8 \cdot S1.2$	Set E9; Reset E8
E10	$E9 \cdot S2.2$	Set E10; Reset E9
E11	$E8 \cdot S2.2$	Set E11; Reset E8
E12	$E11 \cdot S1.2$	Set E12; Reset E11

Lógica programável

- Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET

- Exercício de Aplicação Nº 3 – Solução usando linguagem de contactos (I)

Equações das etapas - Endereços

E0	$SM0.1 + ((M1.2+M1.4) \cdot I0.3 \cdot I0.5)$	Set M0.0; Reset M1.2; Reset M1.4
E1	$M0.0 \cdot I0.0 \cdot \overline{I0.6} + M1.0 \cdot (I0.0 + \overline{I0.1}) \cdot \overline{I0.6}$	Set M0.1; Reset M0.0; Reset M1.0
E2	$M0.1 \cdot T37$	Set M0.2; Reset M0.1
E3	$M0.2 \cdot I0.2$	Set M0.3; Reset M0.2
E4	$M0.3 \cdot I0.4$	Set M0.4; Reset M0.3
E5	$M0.2 \cdot I0.4$	Set M0.5; Reset M0.2
E6	$M0.5 \cdot I0.2$	Set M0.6; Reset M0.5
E7	$(M0.4 + M0.6) \cdot I0.2 \cdot I0.4 \cdot \overline{I0.6}$	Set M0.7; Reset M0.4; Reset M0.5
E8	$M0.7 \cdot T38 \cdot \overline{I0.6}$	Set M1.0; Reset M0.7
E9	$M1.0 \cdot I0.3$	Set M1.1; Reset M1.0
E10	$M1.1 \cdot I0.5$	Set M1.2; Reset M1.1
E11	$M1.0 \cdot I0.5$	Set M1.3; Reset M1.0
E12	$M1.3 \cdot I0.3$	Set M1.4; Reset M1.3

Lógica programável

- **Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET**

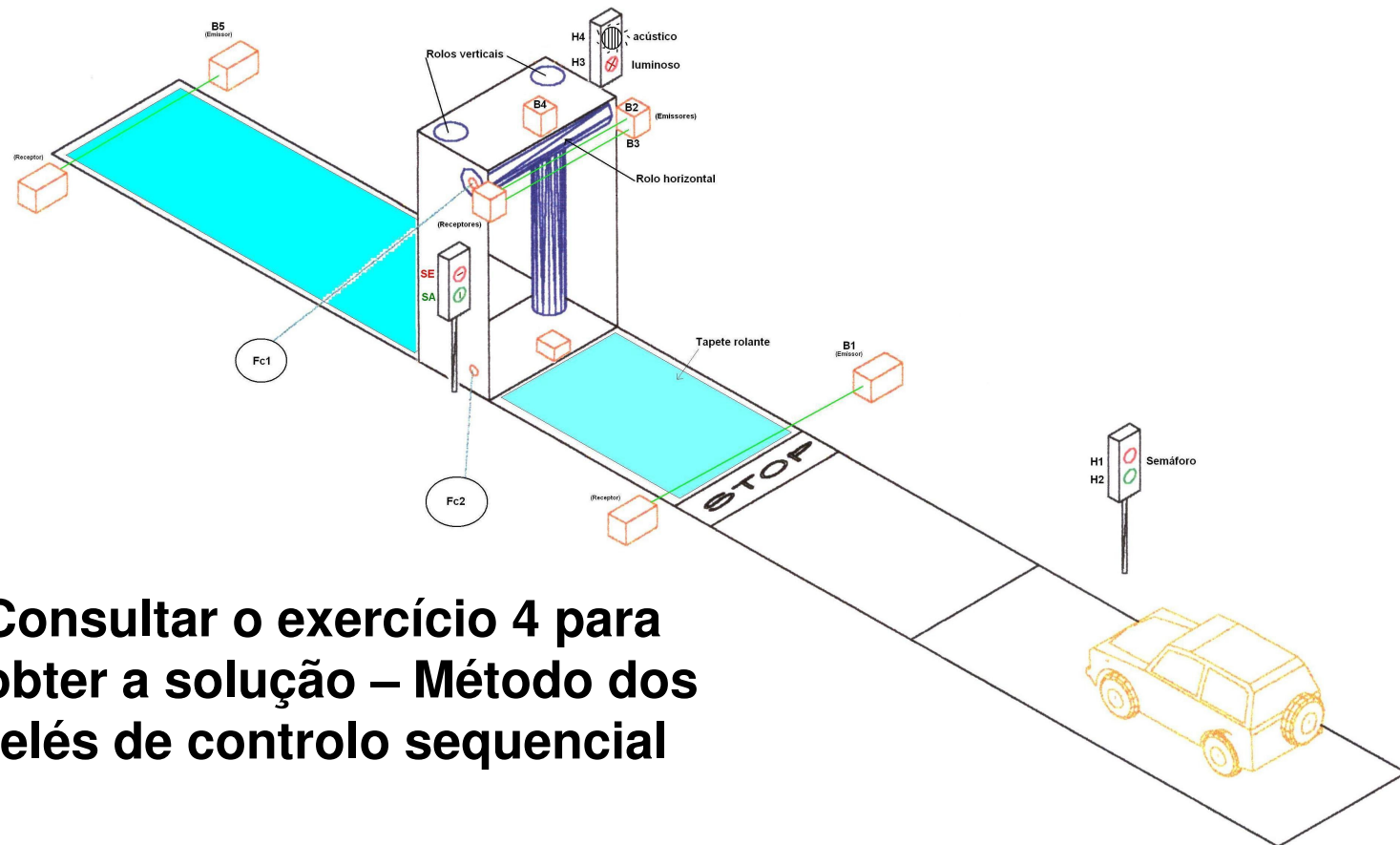
➤ **Exercício de Aplicação Nº 3 – Solução usando linguagem de contactos (I)**



Consultar o exercício 3 (Step7/Micro) para compreender a programação do autómato programável

Lógica programável

- Programação em SFC – Diagramas Funcionais Sequenciais ou GRAFCET - **Exercício de Aplicação Nº 4 – Lavagem automática**



Consultar o exercício 4 para obter a solução – Método dos relés de controlo sequencial