

# Sistemas Embarcados

PROF. JOSENALDE OLIVEIRA

TADS UFRN

[josenalde@eaj.ufrn.br](mailto:josenalde@eaj.ufrn.br)

<https://github.com/josenalde/embeddedsystems>

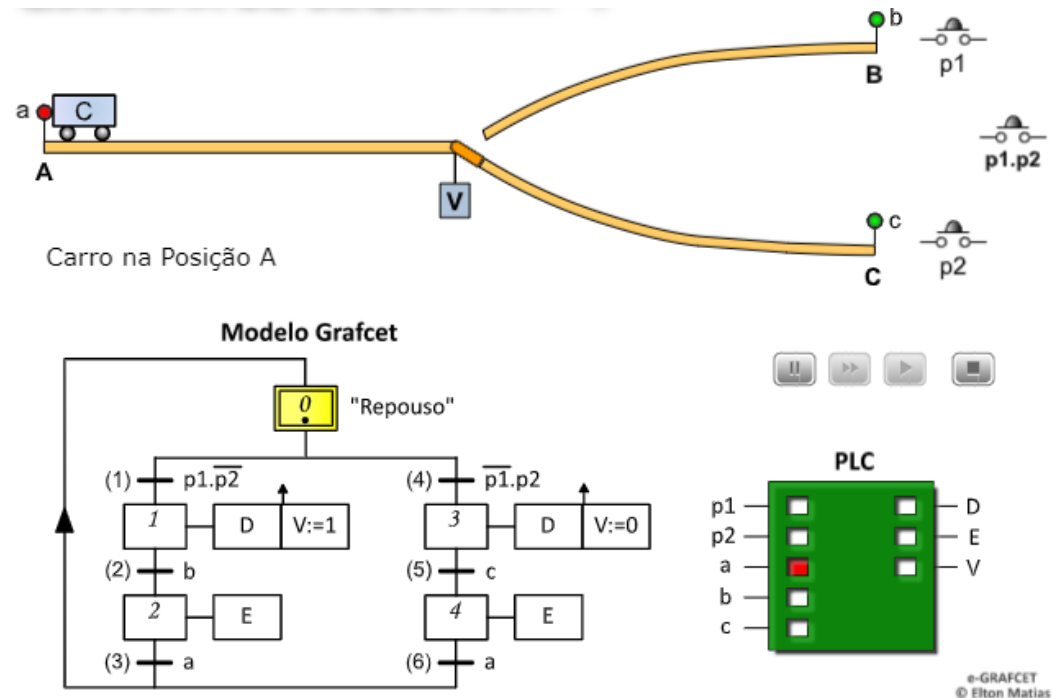
# Modelagem de processos sequenciais: GRAFCET

- Método gráfico para projetos de sistemas automatizados sequenciais (sugestão de vídeo aula: [GRAFCET - Parte I \( Em Português \) – YouTube](#))
- Embora existam CLPs que permitam programar diretamente em GRAFCET (EN 60848), é mais conhecido como ferramenta de projeto, e o LADDER como uma linguagem de programação (portanto existem métodos de conversão GRAFCET-LADDER)
- Em computação, seria equivalente à análise do sistema, documentar solução

“**GR**Aphe **F**onctionnel  
de **C**ommande **E**tapes/**T**ransitions.”

**Ideia:** A partir de etapas/condições iniciais, ocorre transição de etapas, onde em cada etapa uma ou mais ações são efetuadas. As transições possuem regras de disparo (lógica, temporização etc.)

*Numa etapa pode-se ter um vetor de saídas (com seus estados 0, 1)*



# Relembrando...

- Máquina de estado
- Tabela de transições de estado

Table 3.2 Six states of TLC controller

State	North-South LEDs	East-West LEDs	Delay in seconds
0	Green	Red	15
1	Yellow	Red	3
2	Red	Red	3
3	Red	Green	15
4	Red	Yellow	3
5	Red	Red	3

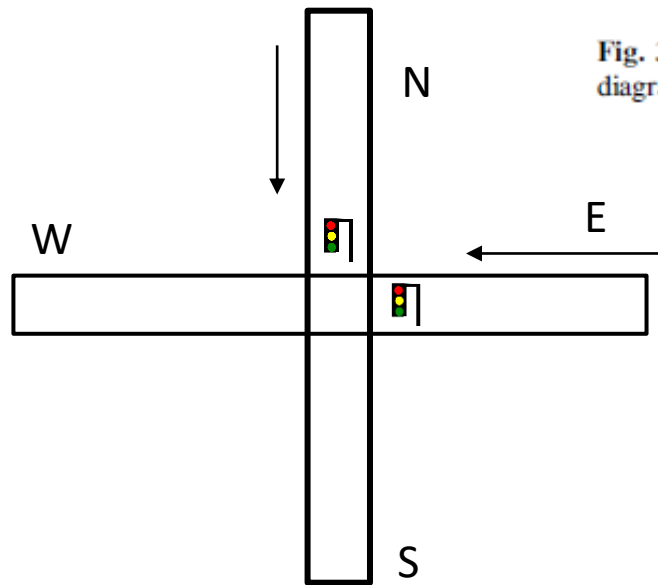
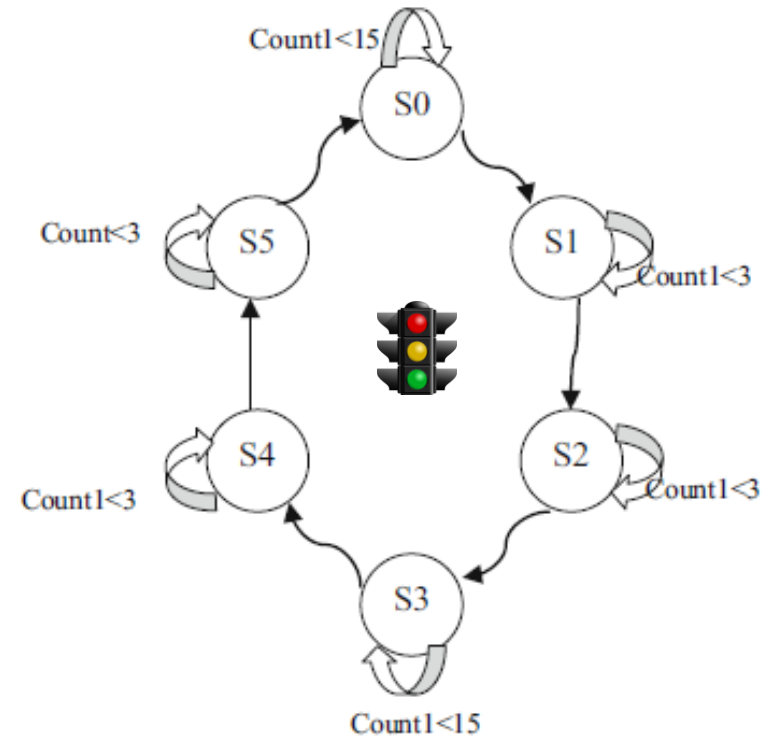


Fig. 3.21 State machine diagram



[https://github.com/josenalde/embeddedsystems/blob/master/src/traffic\\_light.vhd](https://github.com/josenalde/embeddedsystems/blob/master/src/traffic_light.vhd)

# Modelagem de processos sequenciais: GRAFCET

The screenshot shows the GRAFCET Studio interface with the following components highlighted by callouts:

- Tool Bar:** Located at the top, containing icons for file operations (Open, Save, New, Undo, Watch, Zoom on/off), simulation (Overview On/Off, Grid On/Off), and search (Search Ctrl+F).
- Grafcet Pages:** A sidebar on the left showing a list of pages (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100).
- Grafcet Items:** A sidebar on the left showing a list of items (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100).
- Drawing area:** The central workspace where the GRAFCET diagram is created. It shows a sequence of steps (Start, 1, 2, Schleife) with associated actions and transitions.
- I/O Panel:** A sidebar on the right showing a list of I/O variables (B1\_X\_VordereEndlage, B2\_X\_ReferenzPos, B3\_X\_HintereEndlage, B4\_Y\_VordereEndlage, B5\_Y\_ReferenzPos, B6\_Y\_HintereEndlage, B7\_Z\_ObereEndlage, B8\_Z\_ReferenzPos, B9\_Z\_UntereEndlage, B11\_MagnetIsOn, S1\_Start, S2\_Stop, S3\_ACK, S4\_REF, S6\_1\_MAN, S6\_2\_Auto, M1\_X\_PositioningDone).
- Overview:** A small thumbnail view of the entire diagram located in the bottom right corner of the drawing area.
- Symbolic Table:** A table at the bottom of the interface listing symbols and their addresses.

Group	Type	Symbol	Comment	Address	DB address	Flags	Used
Sensor	BOOL	B1_X_VordereEndlage		I0.0	D0.0	{{%0.0}}	0
Sensor	BOOL	B2_X_ReferenzPos		I0.1	D0.1	{{%0.1}}	0
Sensor	BOOL	B3_X_HintereEndlage		I0.2	D0.2	{{%0.2}}	0
Sensor	BOOL	B4_Y_VordereEndlage		I0.3	D0.3	{{%0.3}}	0

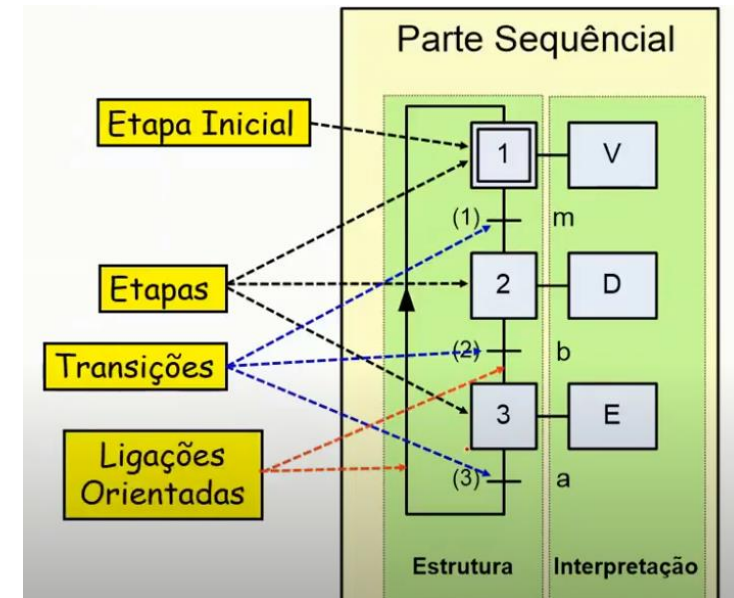
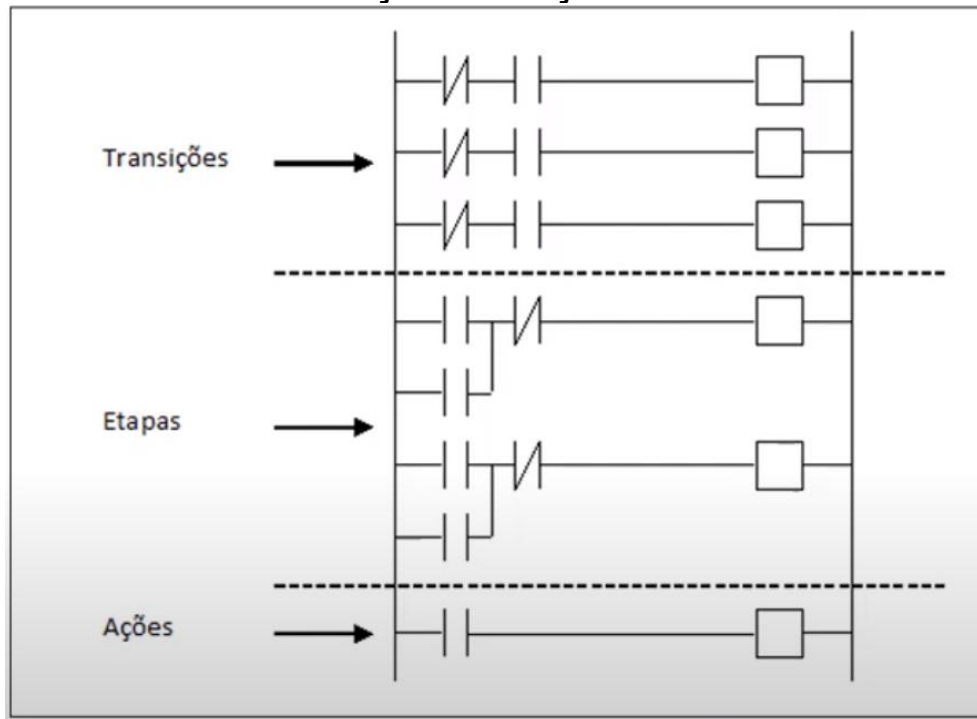
Exemplo de sw para design/simulação: [grafcet studio](https://grafcet-studio.com/), com deploy (upload) do grafcet **diretamente** para CLPs compatíveis (s7-300/400, 1200/1500, codesys v3, raspberry, Arduino Due); Existem outros projetos, inclusive OPEN, com CLPs em Raspberry, Arduino, ESP8266: [Conheça o OpenPLC - O primeiro CLP de Código Aberto Padronizado \(embarcados.com.br\)](https://embarcados.com.br/)

# Conversão GRAFCET – LADDER - Metodologia



- Para CLP baseado em FPGA, não temos conhecimento de ferramentas de geração automática de conversão grafcet – blocos lógicos, ou grafcet-ladder com blocos lógicos, sendo uma oportunidade de desenvolvimento!

- 1) Na estrutura do grafcet temos três elementos básicos: **transições, etapas** e as **ações**
- 2) No Ladder também teremos três partes distintas, nesta ORDEM: transições, depois sequenciamento das etapas e por último a execução das ações



# Conversão GRAFCET – LADDER - Metodologia

## Exemplo 1: [egrafcet.utad.pt/Home/Exemplos\\_Ex1](http://egrafcet.utad.pt/Home/Exemplos_Ex1)

Considere um carro que se pode movimentar entre as posições A e B.

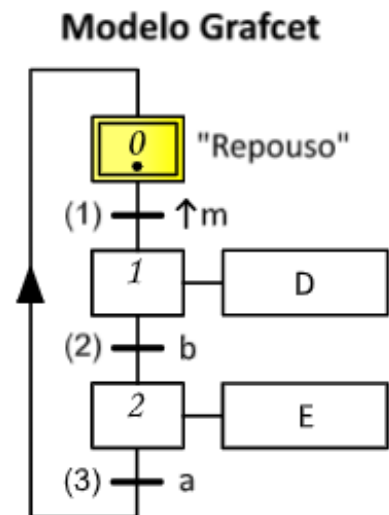


A presença do carro em A é representada pela variável Booleana  $a=1$  e em B é representada por  $b=1$ . O carro é comandado por um botão  $m$  ( $m=1$  quando pressionado). Inicialmente o carro está na posição A (repouso). As saídas do sistema de controlo são D e E ( $D=1$ , o carro desloca-se para a direita;  $E=1$ , o carro desloca-se para a esquerda). Com a utilização de Grafcets vão ser implementados os controladores lógicos que correspondem às seguintes situações:

Sensibilidades/ Entradas	Variável	Saídas	Variável
Botão de acionamento	M (I0)	Carro vai para a direita	D (Q0)
Fim de curso A	A (I1)	Carro vai para a esquerda	E (Q1)
Fim de curso B	B (I2)		

## Memorização (variáveis internas) das etapas e transições

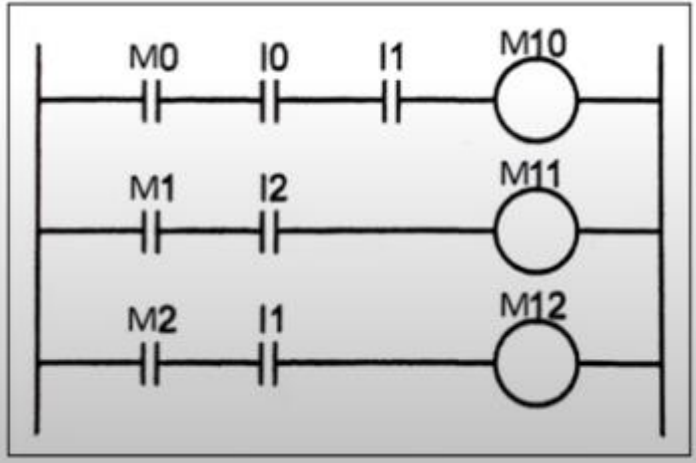
Transição	Variável	Etapas	Variável
E0 – E1 (1)	M10	0	M0
E1 - E2 (2)	M11	1	M1
E2 - E0 (3)	M12	2	M2



# Conversão GRAFCET – LADDER - Metodologia

**Exemplo 1:** [egrafcet.utad.pt/Home/Exemplos\\_Ex1](http://egrafcet.utad.pt/Home/Exemplos_Ex1)

Transições: etapa acima ATIVA e receptividade VERDADEIRA



Etapa acima da transição

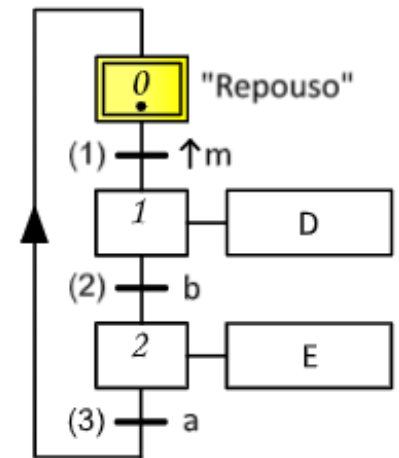
$M10 = M0 \text{ AND } (I0 \text{ AND } I1)$

Receptividade (condição)

$M11 = M1 \text{ AND } I2$

$M12 = M2 \text{ AND } I1$

Modelo Grafcet

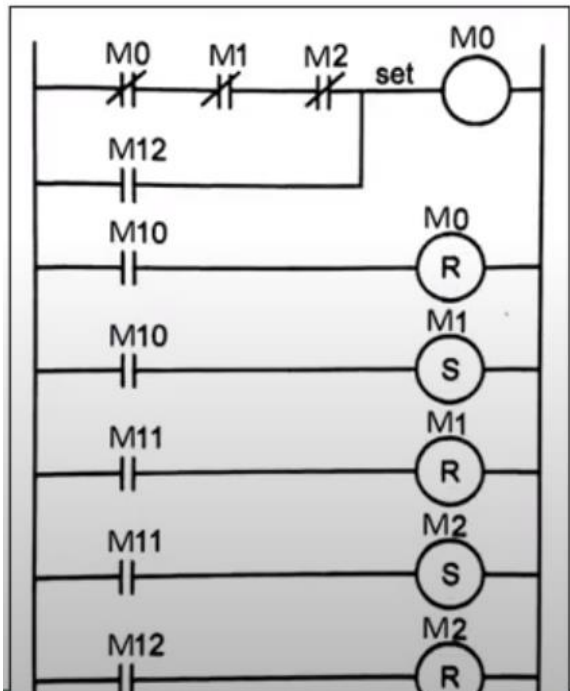


# Conversão GRAFCET – LADDER - Metodologia

**Exemplo 1:** [egrafcet.utad.pt/Home/Exemplos\\_Ex1](http://egrafcet.utad.pt/Home/Exemplos_Ex1)

Etapas (estados): a etapa pode necessitar ficar ativa por mais tempo que a transição que a ativa, normalmente pulsante. Logo existem soluções com SET/RESET e com SELO

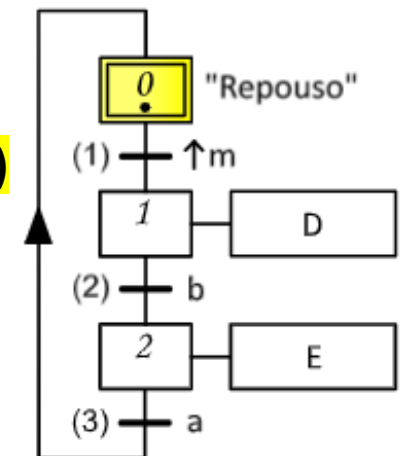
## LÓGICA SET/RESET



Qual transição ativa etapa 0 (M0)? = M12 (2->0)  
Outra condição INICIAL é não haver etapas ativas, para então SETAR M0 como INICIAL, logo, faz-se a conjunção **NOT(M0) AND NOT(M1) AND NOT(M2)**

Transição que faz chegar em Etapa e Transição que faz sair da Etapa

## Modelo Grafcet



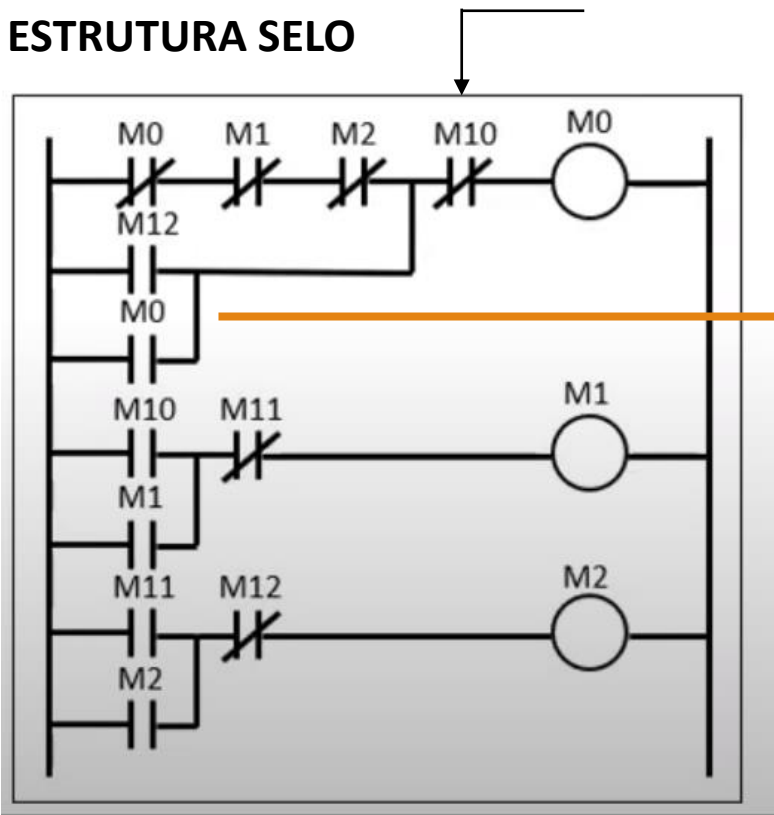


# Conversão GRAFCET – LADDER - Metodologia

**Exemplo 1:** [egrafcet.utad.pt/Home/Exemplos\\_Ex1](http://egrafcet.utad.pt/Home/Exemplos_Ex1)

Etapas (estados): a etapa pode necessitar ficar ativa por mais tempo que a transição que a ativa, normalmente pulsante. Logo existem soluções com SET/RESET e com **SELO**

## ESTRUTURA SELO



Este AND com a transição 0->1 NF, indica que se a transição ocorrer (M10=1, logo NOT(M10)=0), M0 = 0, como se fosse um botão de STOP. Este padrão repete-se para as outras etapas, sempre com a transição de saída da etapa em NF

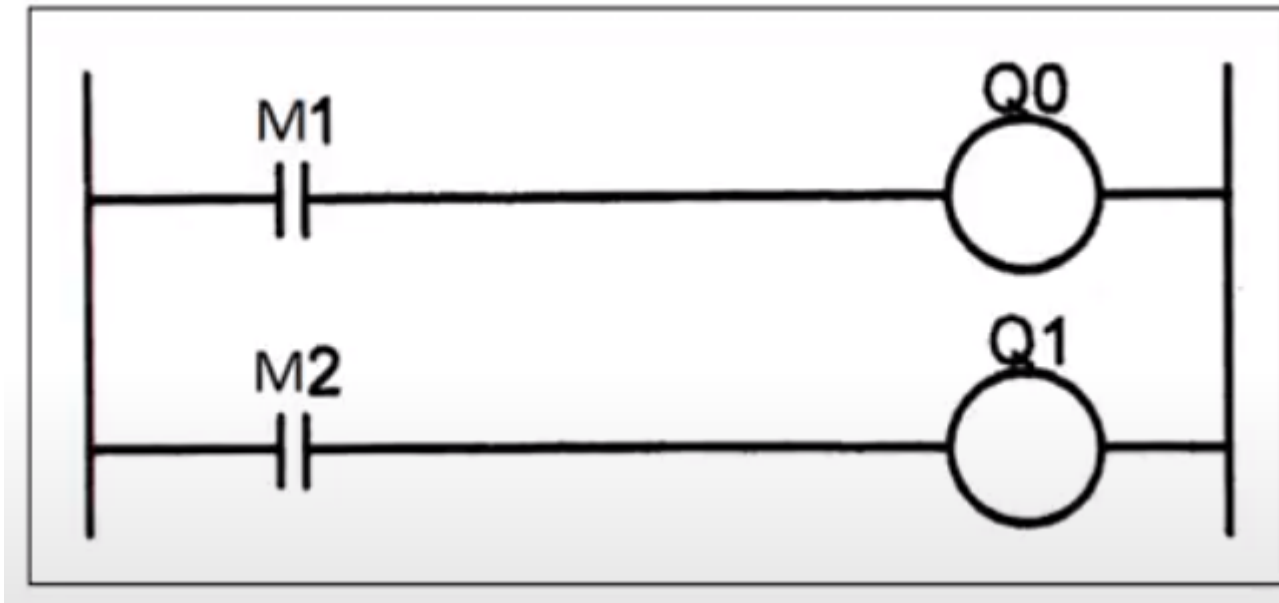
Selo mantém memória (etapa) ativa = 1

Esquema geral:  $\text{memoriaEtapa} = (\text{transicaoChegada} \text{ OR } \text{memoriaEtapa}) \text{ AND NOT } (\text{transicaoSaida})$

# Conversão GRAFCET – LADDER - Metodologia

**Exemplo 1:** [egrafcet.utad.pt/Home/Exemplos\\_Ex1](http://egrafcet.utad.pt/Home/Exemplos_Ex1)

Saídas: as saídas equivalentes às respectivas etapas, quando ativas, são disparadas

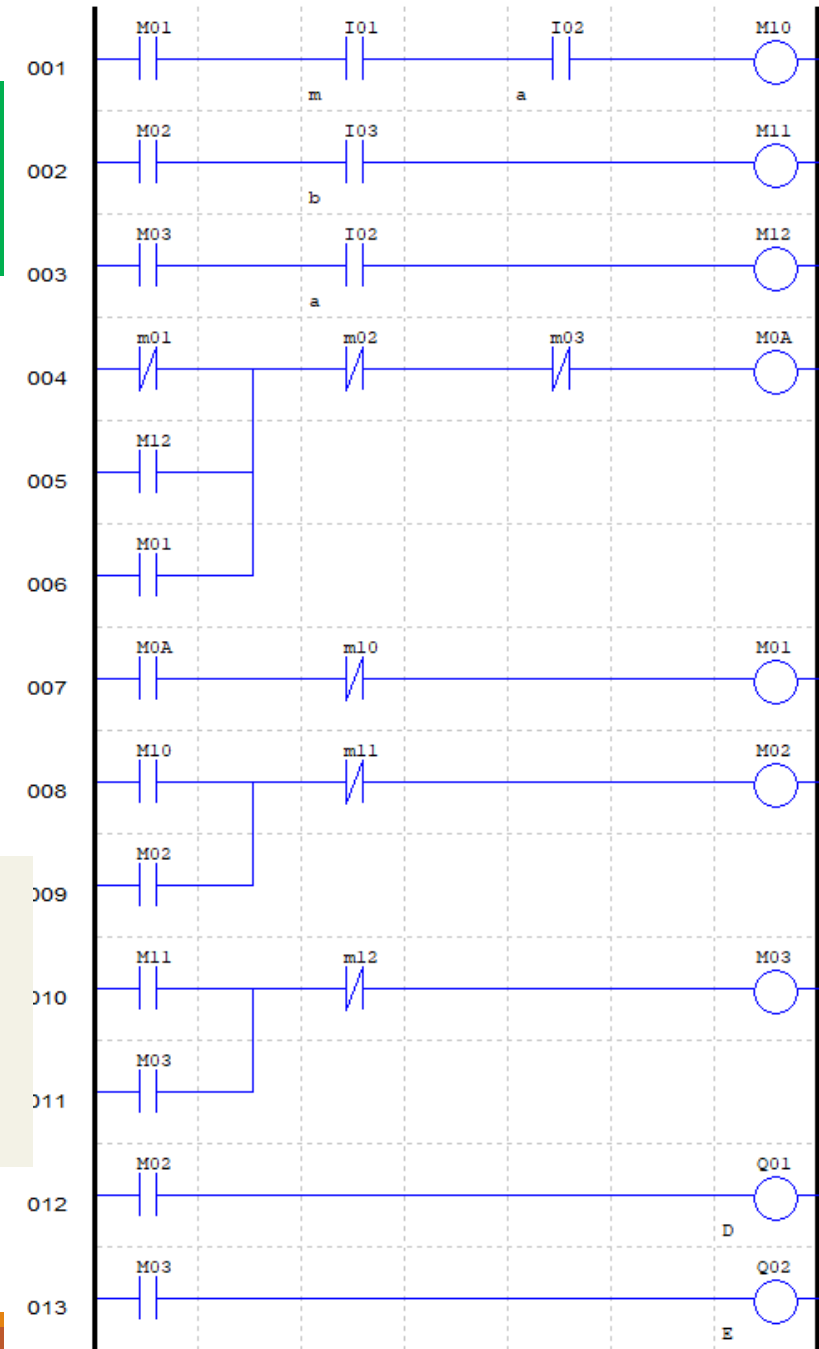
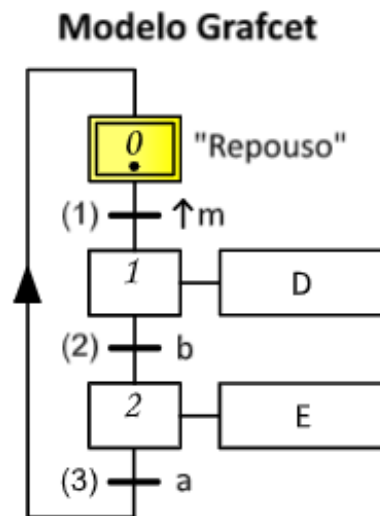


Exercício 1: montagem deste diagrama Ladder no Clic02. Ver exemplo2.cli em [embeddedsystems/ladderdiagrams](https://github.com/josenalde/embeddedsystems) at master · josenalde/embeddedsystems (github.com)

# Conversão GRAFCET – LADDER - Metodologia

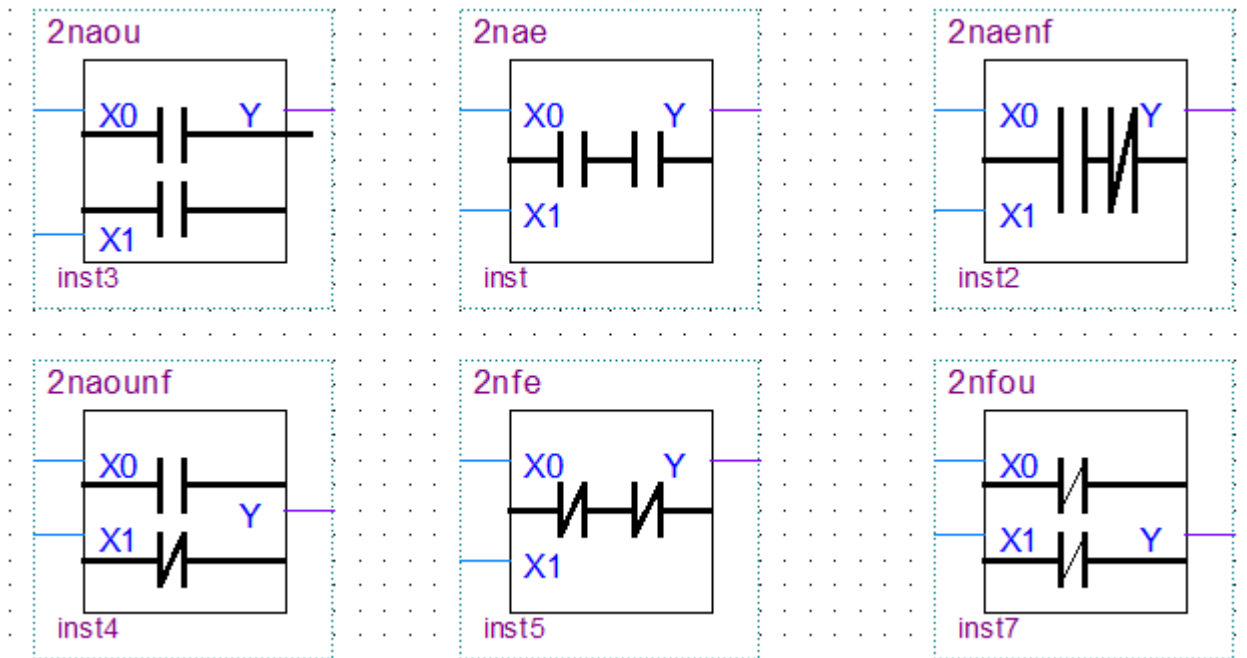
**Exemplo 1:** [egrafcet.utad.pt/Home/Exemplos\\_Ex1](http://egrafcet.utad.pt/Home/Exemplos_Ex1)

Exercício 1: montagem deste diagrama Ladder no Clic02.  
Ver exemplo2.cli em [embeddedsystems/ladderdiagrams](https://github.com/josenalde/embeddedsystems) at  
master · [josenalde/embeddedsystems](https://github.com/josenalde/embeddedsystems) (github.com)



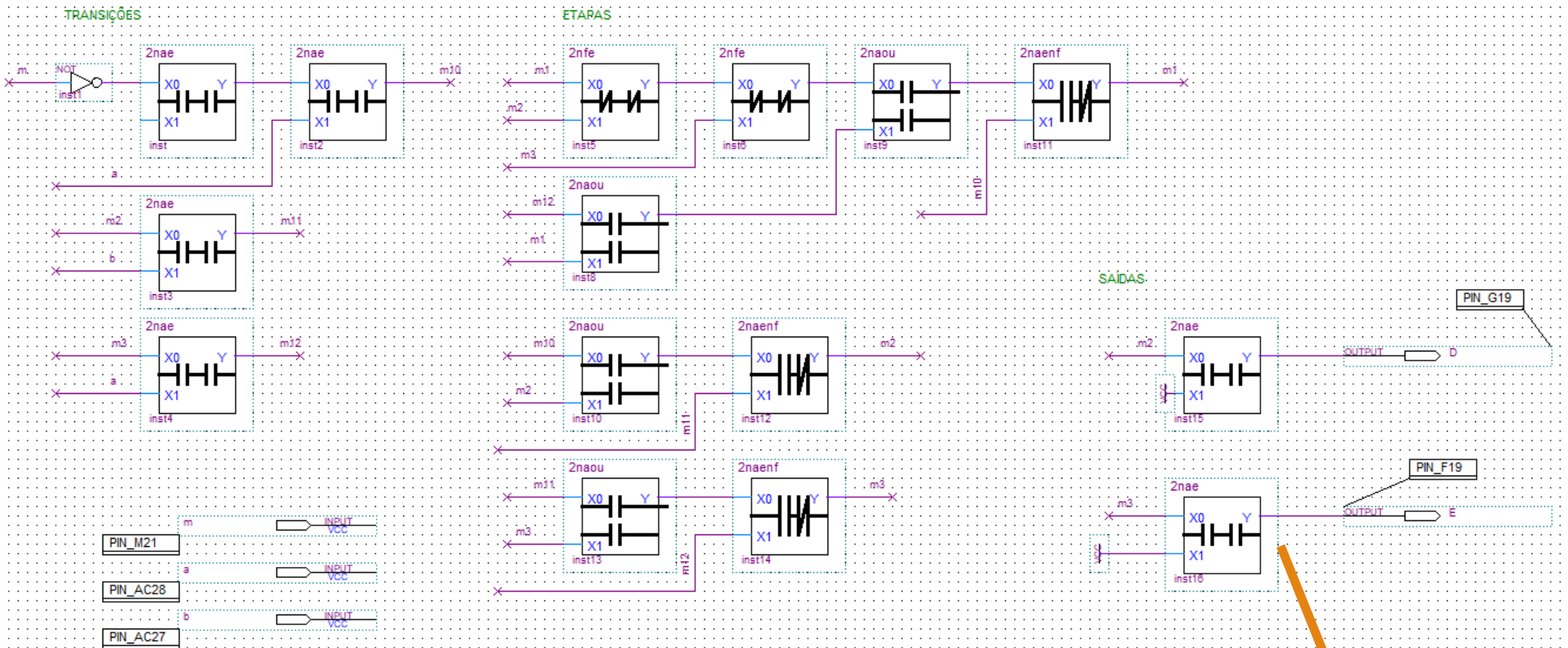
Para escrever esta solução em FPGA, precisamos de blocos lógicos (macrofunções) que representem os relés NA, NF e respectivas combinações AND e OR. Nossa referência propõe os seguintes blocos, mas podemos criar tantos quanto precisarmos

# Bloco (.bsf – block symbol files, com respectivos .bdf (block diagram files) Quartus



Para escrever esta solução em FPGA, precisamos de blocos lógicos (macrofunções) que representem os relés NA, NF e respectivas combinações AND e OR. Nossa referência propõe os seguintes blocos, mas podemos criar tantos quanto precisarmos. Uma opção é colocar tais arquivos .bdf e .bsf numa pasta com nome sugestivo, como lib ou include, e referenciar na criação do projeto

# Projeto carro1



$$Y = X0.1 = X0 \text{ (NA)}$$

$$E = m3.1 = m3$$

Sensibilidade/ Entradas	Variável	Chave PLAC A	PINO DE2- 115	PINO DE2
Botão de acionamento	M (I0)	KEY[1]	M21	N23
Fim de curso A	A (I1)	SW[1]	AC28	N26
Fim de curso B	B (I2)	SW[2]	AC27	P25

Saídas	Variável	LED	PINO DE2- 115	PINO DE2
Carro vai para a direta	D (Q0)	LEDR[0]	G19	AE23
Carro vai para a esquerda	E (Q1)	LEDR[1]	F19	AF23