Controladores Lógicos Programáveis

Eletrotécnica Geral

Depto. de Engenharia de Energia e Automação Elétricas
Escola Politécnica da USP

21 de novembro de 2016

Histórico

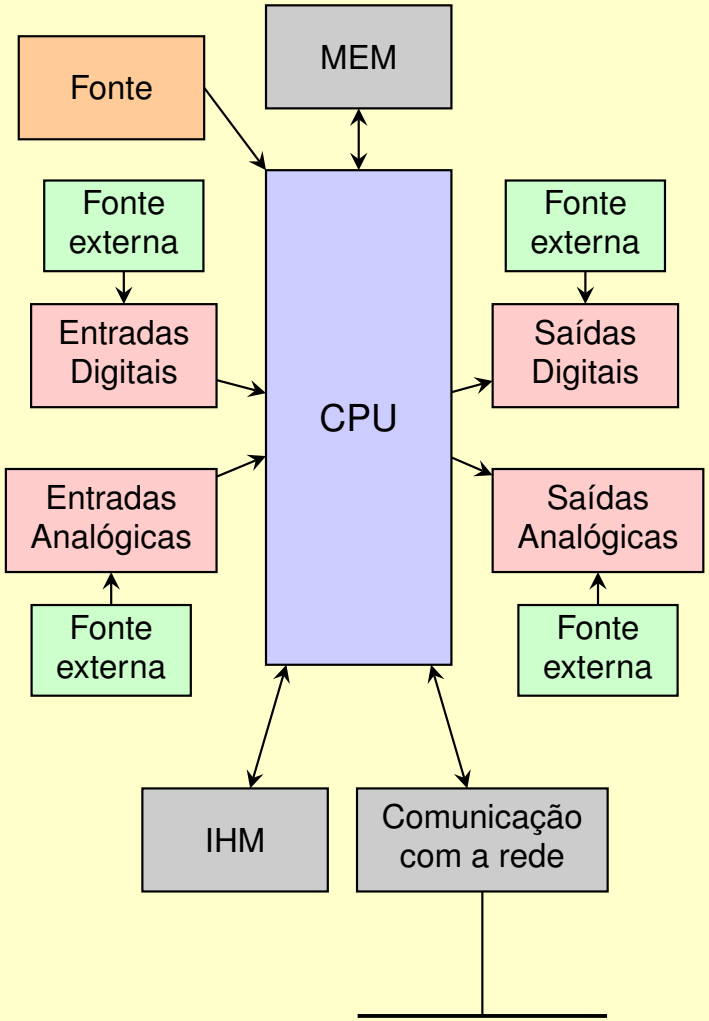
- *"O desenvolvimento do CLPs (Controladores Lógicos Programáveis) ou CPs (Controladores Programáveis) começou por volta de 1968 quando a General Motors solicitou à indústria eletrônica uma alternativa para a lógica eletromecânica baseada em relés."* (PEREIRA, 2003)
- *"Os sistemas de relés utilizados nas atividades industriais (montagem, carregamento e controle de máquinas) haviam se tornados grandes e complexos, aumentando significativamente os custos de manutenção e baixando a confiabilidade."* (PEREIRA, 2003)
- *"Outro problema era a grande complexidade envolvida em qualquer mudança na planta industrial ou produtiva."* (PEREIRA, 2003)
- *"Inicialmente os CLPs foram produzidos somente para simular a ação de relés num circuito de intertravamento. Hoje, também incorporam funções avançadas como: controle estatístico, controle de malha, comunicação em rede, entre outras."* (PEREIRA, 2003)

Arquitetura

- O CLP é um equipamento eletrônico digital que *"pode ser programado para executar instruções que controlam dispositivos, máquinas e operações de processos, por meio da implementação de funções específicas como lógica de controle, sequenciamento, controle de tempo, operações aritméticas, transmissão de dados*", entre outras. (PEREIRA, 2003)
- Os CLPs são bastante adequados para operação em ambientes severos, sujeitos, por exemplo, a altas temperaturas, vibrações, ruídos elétricos e poluição atmosférica.
- Os fabricantes de CLP oferecem uma grande variedade de modelos, que diferem quanto ao número de entradas e saídas, memória, conjunto de instruções, velocidade de processamento, conectividade, flexibilidade, interface humano-máquina, protocolos de comunicação, entre outros.

Componentes básicos

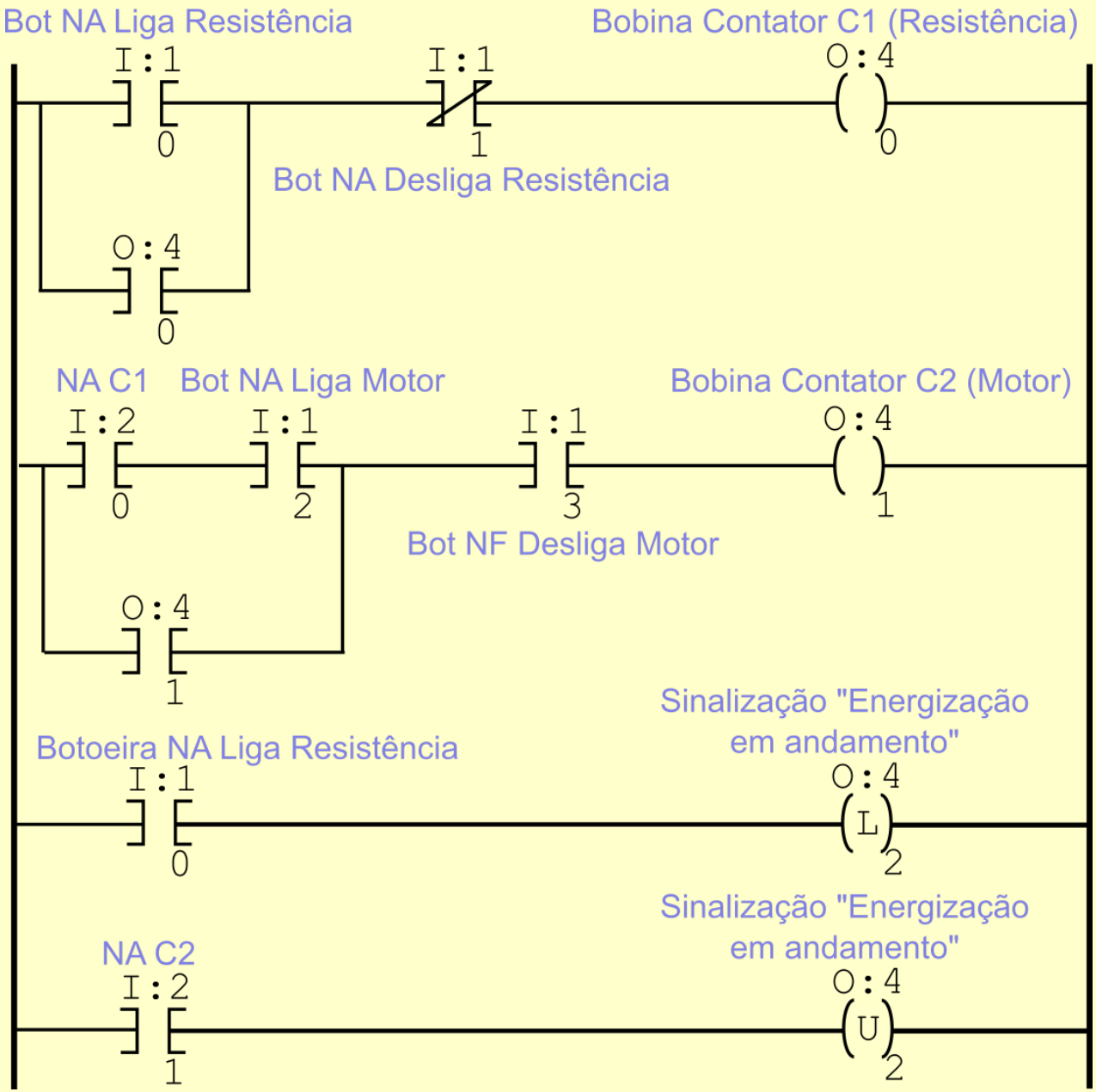
- CPU (Unidade Central de Processamento);
- Memória (volátil e não volátil);
- Módulos de entrada e saída (I/O: inputs/outputs)
  - quantidade e tipos de módulos de acordo com a necessidade do usuário;
  - entradas e saídas digitais e analógicas;
  - tensão/corrente, CA/CC ...
- IHM (interface humano-máquina)
  - programação, visualização, operação;
- Fonte de alimentação
  - Alimentação do rack e componentes de *hardware*, execução do programa;



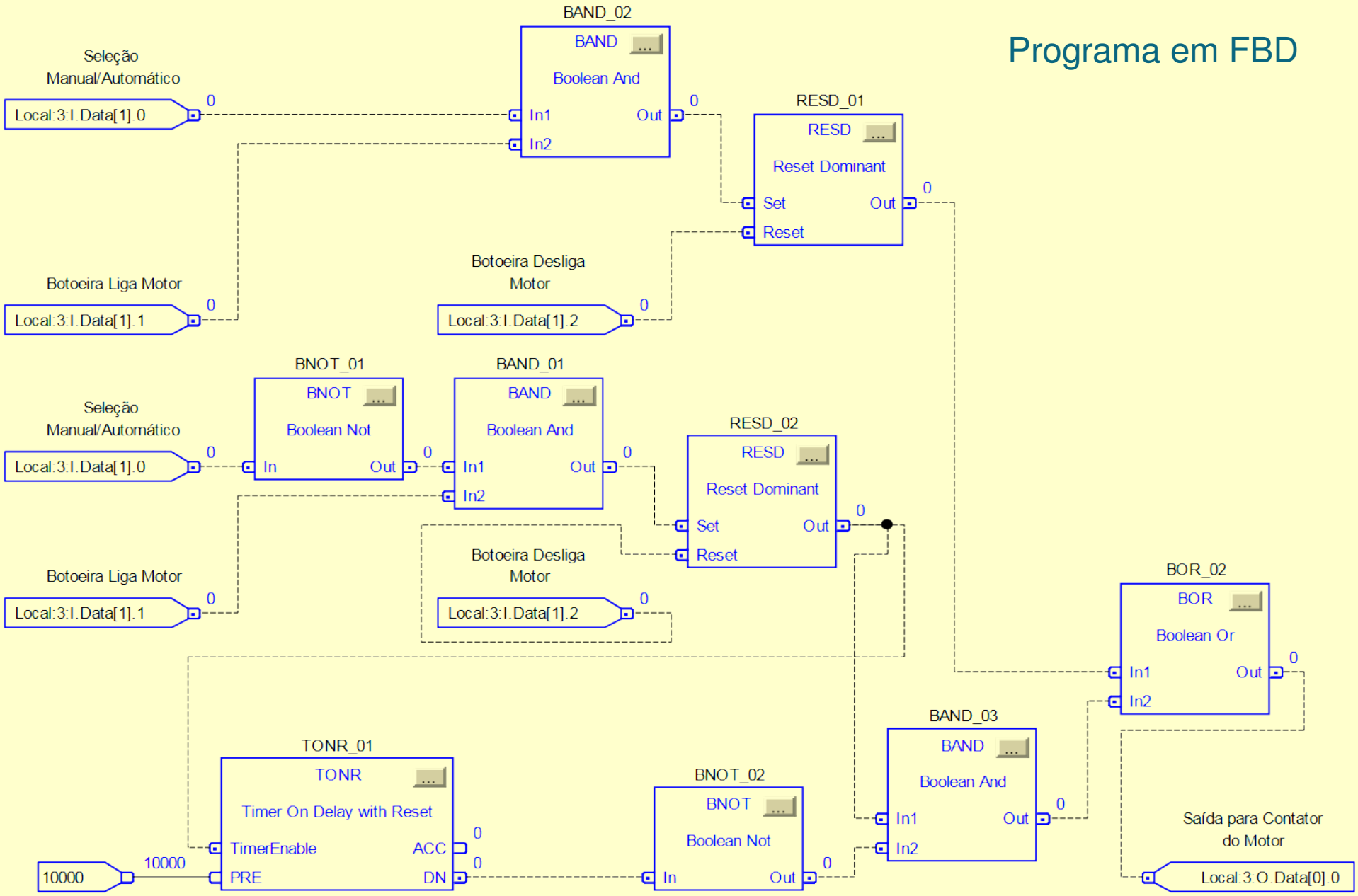
Linguagens de Programação, norma IEC 61131-3

- **Ladder** ("escada"),LD;
- Function Block Diagram, FBD;
- Sequential Function Chart, SFC;
- Structured Text, ST;
- Instruction List, IL.

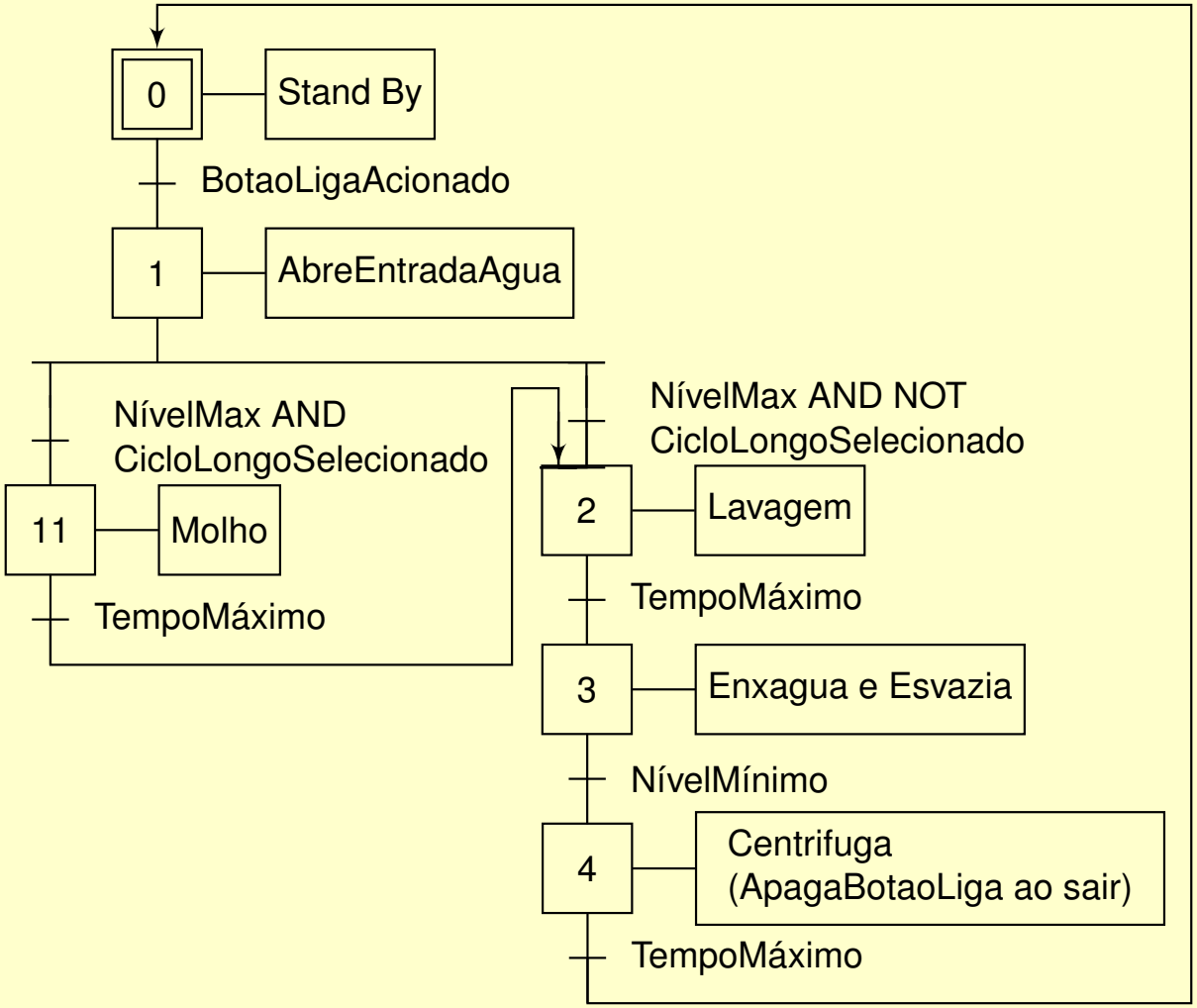
Programa em Ladder



Programa em FBD



Programa em SFC



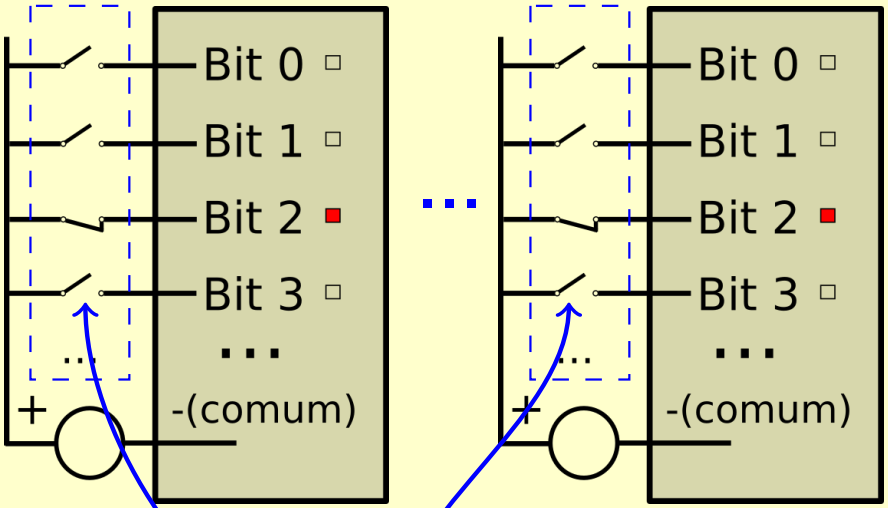
Programa em ST

```
CASE CorPintura OF

0:    TintaCiano      := 0;
      TintaMagenta    := 100;
      TintaYellow     := 100;

1:    TintaCiano      := 0;
      TintaMagenta    := 50;
      TintaYellow     := 100;

2:    TintaCiano      := 100;
      TintaMagenta    := 0;
      TintaYellow     := 100;

END_CASE;
```

Modos de funcionamento do CLP

- Operação (RUN): executa o programa (Ladder, FBD...), monitora dispositivos de entrada, energiza dispositivos de saída;
- Programação (PROG): alteração do programa, saídas são desligadas;
- RUN/PROG selecionados remotamente (REM) ou por comando físico;
- Possibilidade de alteração on-line e de forçar saídas (situações excepcionais!).

Cartões de entrada digitais:

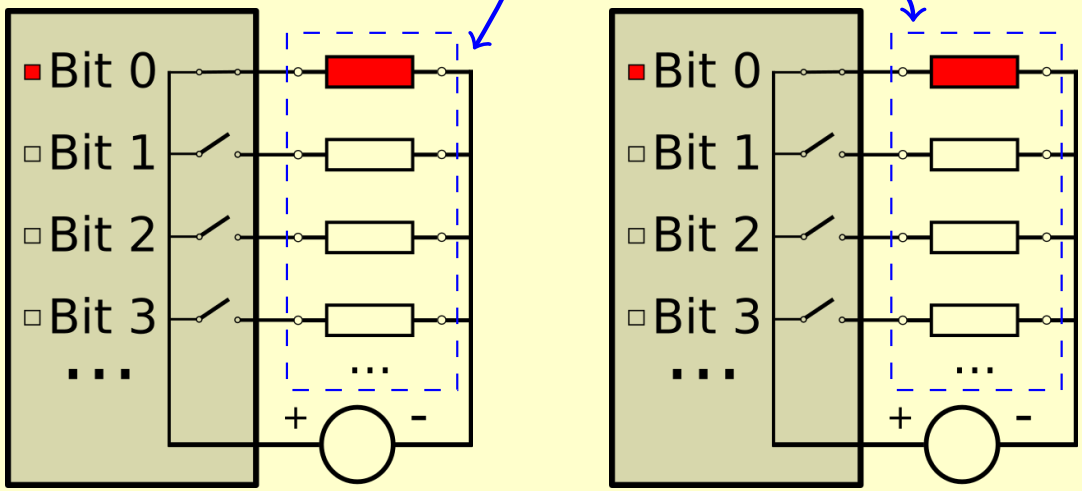


Saídas para o processo

- V+ (=1)
- Vref (=0)
- sinalizações
- contatores
- ...

Contatos do processo

- botoeiras
- chaves
- sensores
- ...



Cartões de saída digitais:

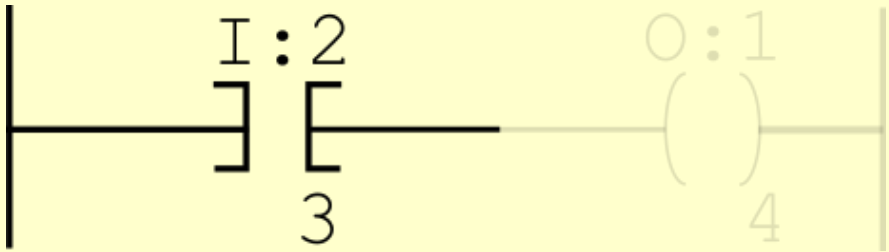
Instruções básicas em linguagem Ladder

- examinar;
- energizar/desenergizar;
- temporizar;
- contar transições;
- operações matemáticas;
- outras, de acordo com o modelo e fabricante.

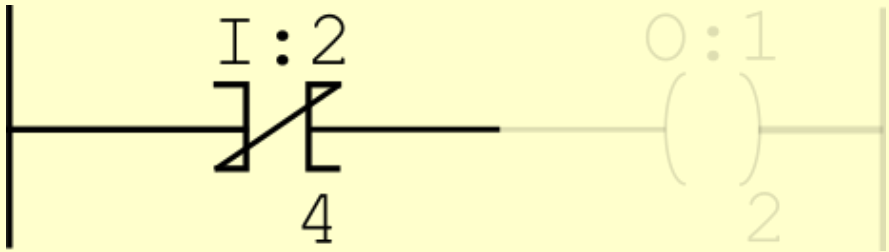
- Execução de programas no CLP:
  - *Scan* (varredura) dos terminais dos cartões de entrada, copiando os seus valores para uma área da memória do CLP;
  - *Scan* das linhas do programa, de forma sequencial, a partir dos valores de entrada na memória, atualizando a área da memória relativa às saídas;
  - *Scan* (varredura) de saída, copiando os valores da memória para os terminais do cartão de saída;
- Apesar dessa linguagem utilizar instruções derivadas de circuitos elétricos baseados em contatos, o seu funcionamento não é idêntico!!

- Tipos básicos de dados (variáveis)
  - Entradas digitais/analógicas;
  - Saídas digitais/analógicas;
  - Variáveis internas digitais/analógicas;
  - Temporizadores, contadores;
  - Strings;
  - Outros...

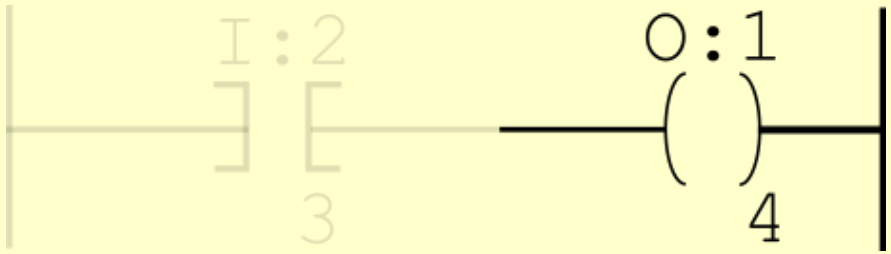
- **Instrução XIC (Examine If Closed)**
- Verifica se o bit em questão está em 1 (entrada ativada = contato do processo fechado)
- O endereçamento desse exemplo se refere ao bit 3 do cartão de entrada (*Input*) #2 (poderia também ser uma variável interna binária)
- Se o bit em questão estiver em 1, a instrução retorna 1 ("contato" lógico fechado)
- Se o bit em questão estiver em 0, a instrução retorna 0 ("contato" lógico aberto)



- **Instrução XIO (Examine If Open)**
- Verifica se o bit em questão está em 0 (entrada desativada = contato do processo aberta)
- O endereçamento desse exemplo se refere ao bit 4 do cartão de entrada (*Input*) #2 (poderia também ser uma variável interna binária)
- Se o bit em questão estiver em 0, a instrução retorna 1 ("contato" lógico fechado)
- Se o bit em questão estiver em 1, a instrução retorna 0 ("contato" lógico aberto)

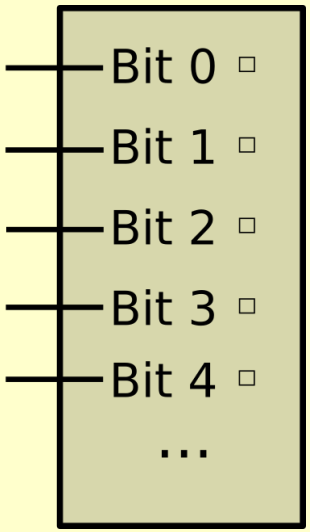


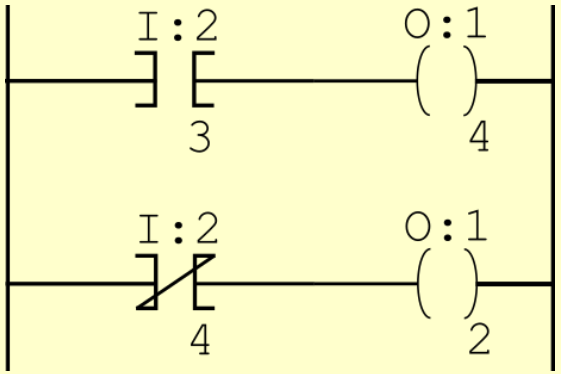
- **Instrução OTE (Output Energize)**
- Caso haja continuidade lógica na linha que chega até essa instrução, energiza o bit em questão
- O endereçamento desse exemplo se refere ao bit 4 do cartão de saída (*Output*) #1 (poderia também ser uma variável interna binária)



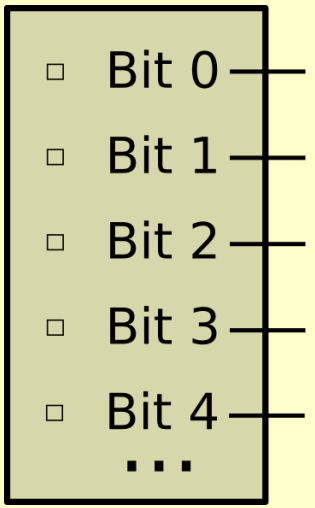
Exemplo bastante simples...

Cartão de entrada #2





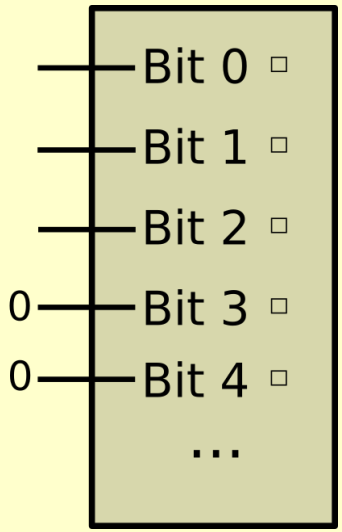
Cartão de saída #1

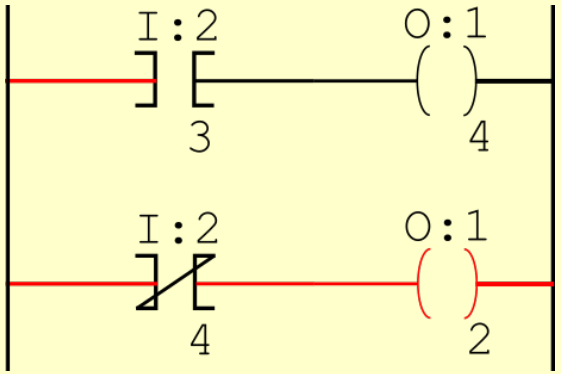


Modo PROG

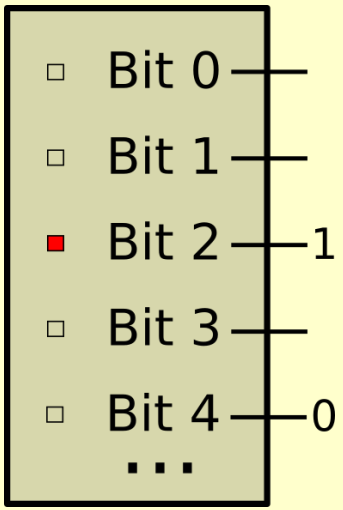
Exemplo bastante simples...

Cartão de entrada #2





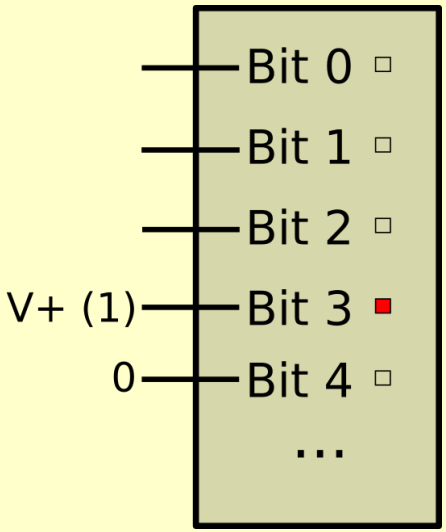
Cartão de saída #1

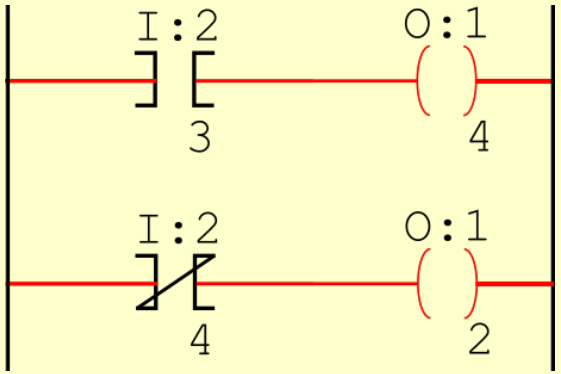


Modo RUN

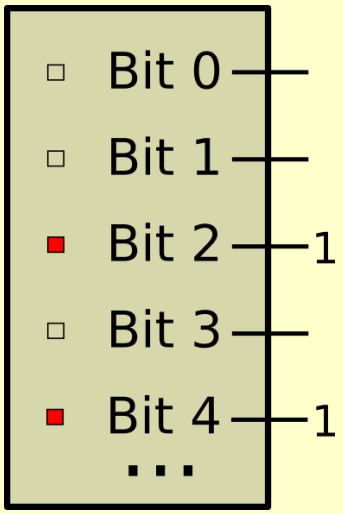
Exemplo bastante simples...

Cartão de entrada #2





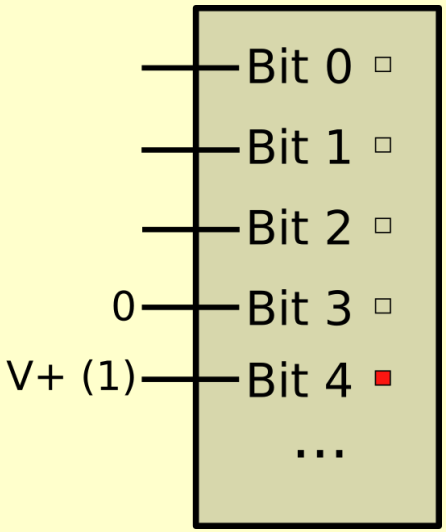
Cartão de saída #1

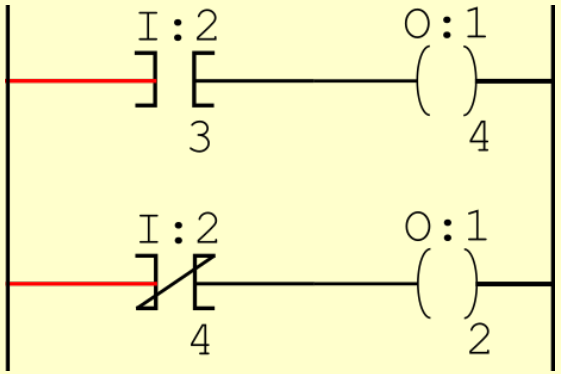


Modo RUN

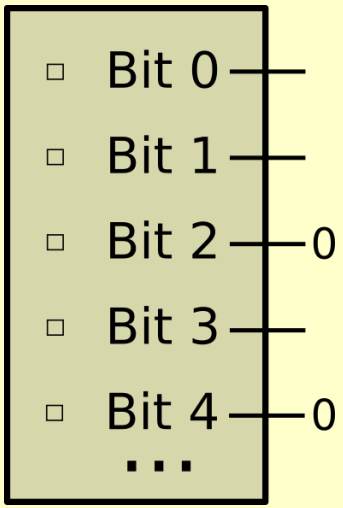
Exemplo bastante simples...

Cartão de entrada #2





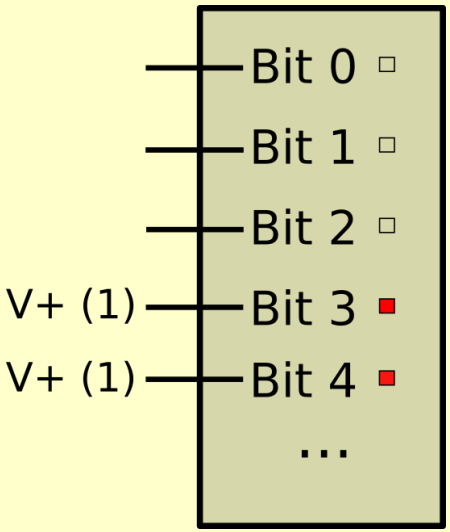
Cartão de saída #1

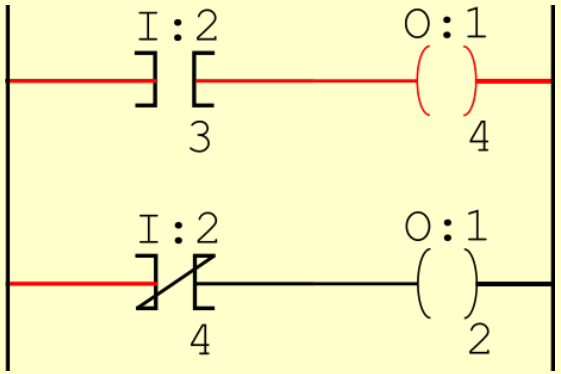


Modo RUN

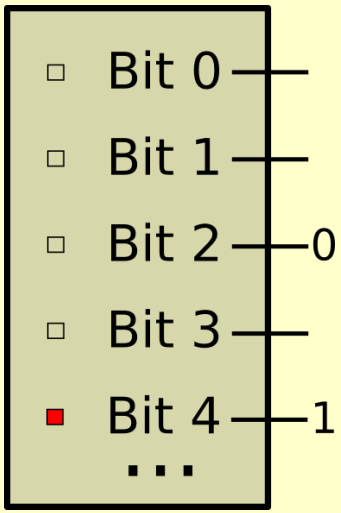
Exemplo bastante simples...

Cartão de entrada #2



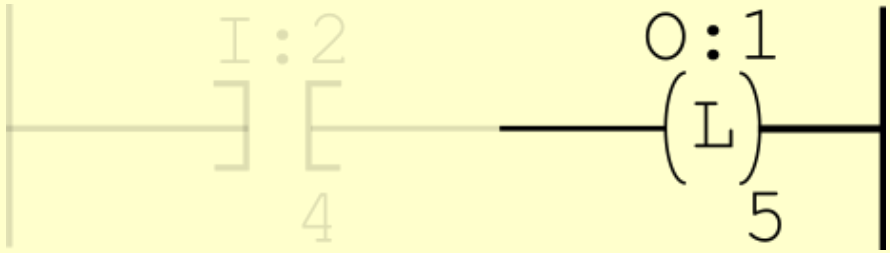


Cartão de saída #1

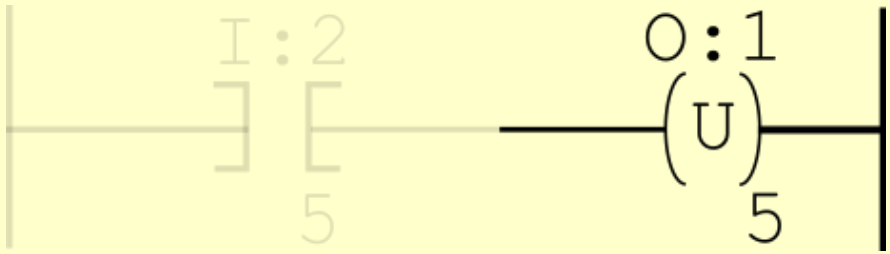


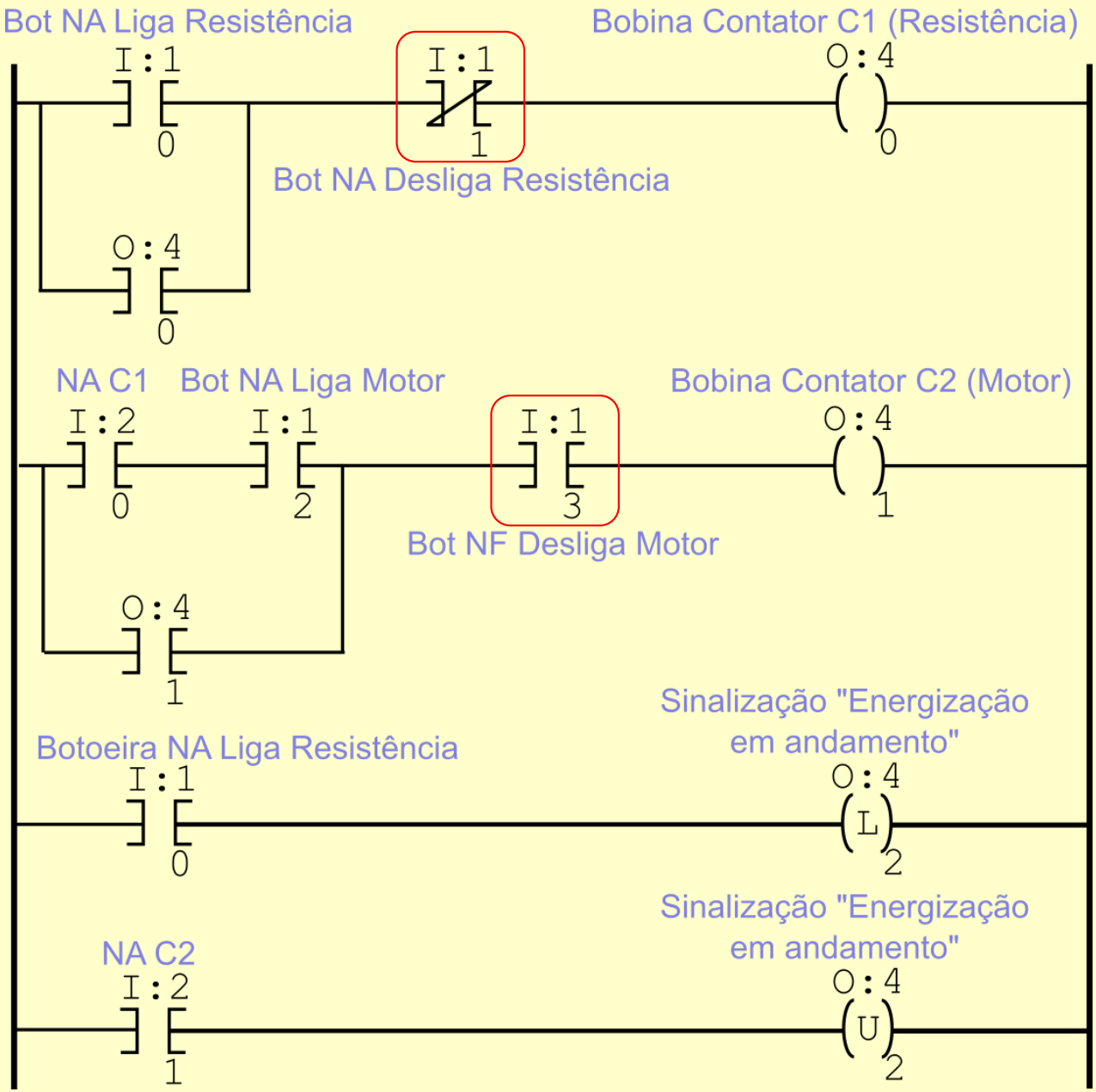
Modo RUN

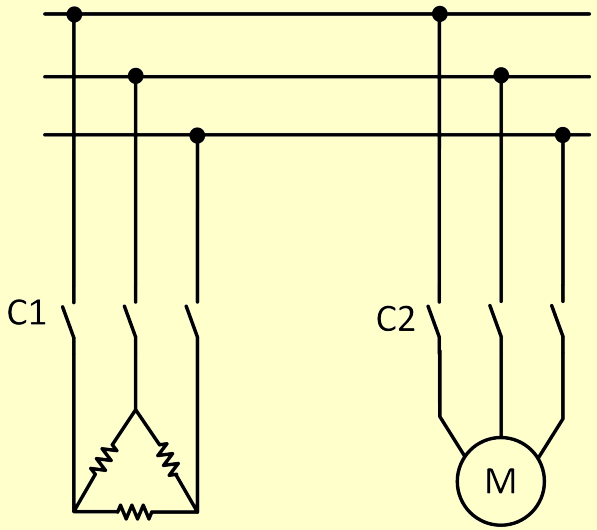
- **Instrução OTL (Output Latch)**
- Caso haja continuidade lógica na linha que chega até essa instrução, energiza o bit em questão;
- Diferentemente do OTE, caso a entrada se altere de 1 para 0, o bit de saída continua energizado (SET).

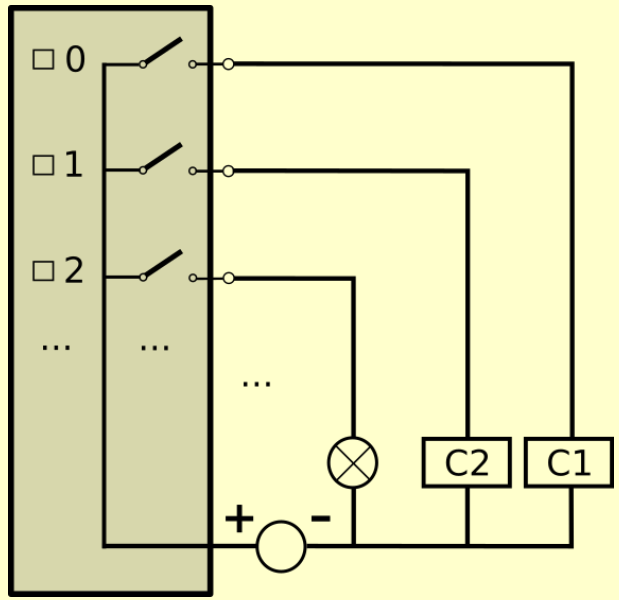


- **Instrução OTU (Output Unlatch)**
- Desenergiza (RESET) uma saída energizada por um OTL (Output Latch)

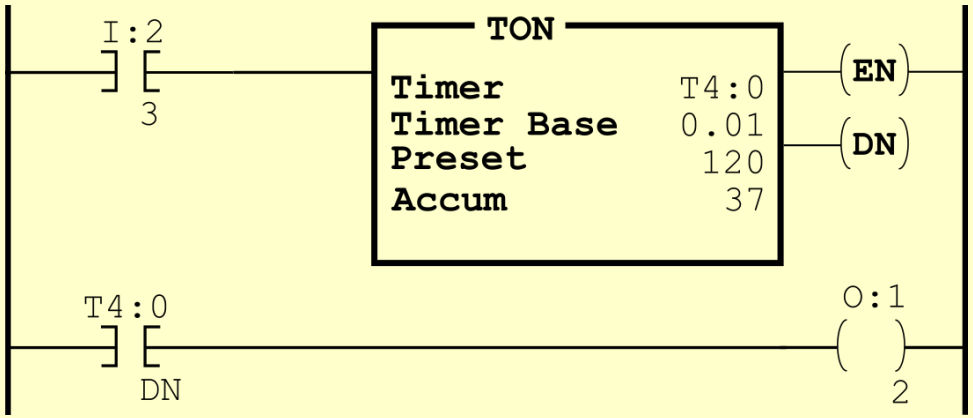



Bot NA Liga Resistência
I:1
0
I:1
1
Bot NA Desliga Resistência
Bobina Contator C1 (Resistência)
O:4
0
O:4
0
NA C1
Bot NA Liga Motor
I:2
0
I:1
2
I:1
3
Bot NF Desliga Motor
Bobina Contator C2 (Motor)
O:4
1
O:4
1
Botoeira NA Liga Resistência
I:1
0
Sinalização "Energização em andamento"
O:4
L
2
NA C2
I:2
1
Sinalização "Energização em andamento"
O:4
U
2

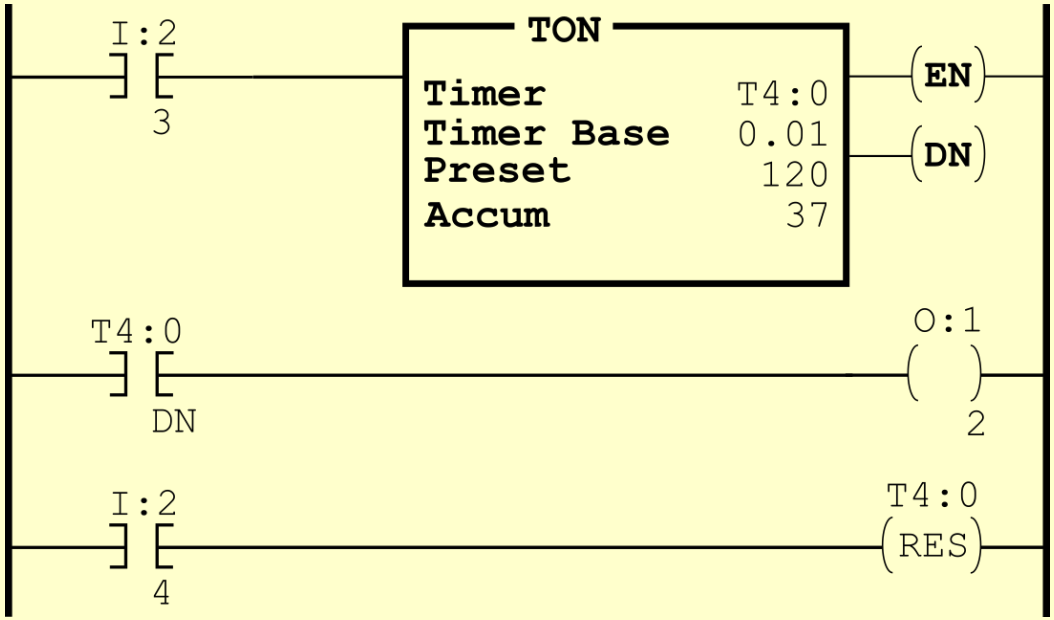

C1
C2
M


0
1
2
C2
C1
+
-

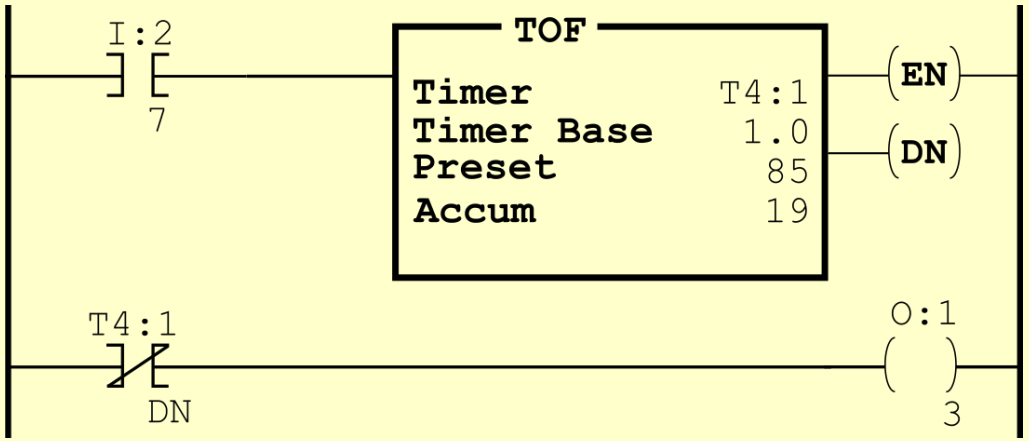
- **Instrução TON (Timer On-Delay)**
- Conta o tempo a partir do instante em que a condição da linha se torna verdadeira (atraso na energização);
- Identificação (exclusiva) do timer (T4:n)
- Ajuste de tempo nos parâmetros *Preset* e *Timer Base* (1 s; 0,01 s; 0,001 s);
- Bit 15 (EN=Enable): Linha habilitada = na condição lógica 1;
- Bit 14 (TT=Timer Timing): Temporização em andamento; linha está verdadeira mas ainda não atingiu tempo ajustado;
- Bit 13 (DN=Done): Se temporização já atingiu tempo ajustado, 0 $\rightarrow$ 1;
- Valor atual de temporização (*Accum*) pode ser lido no valor inteiro T4:n.ACC.

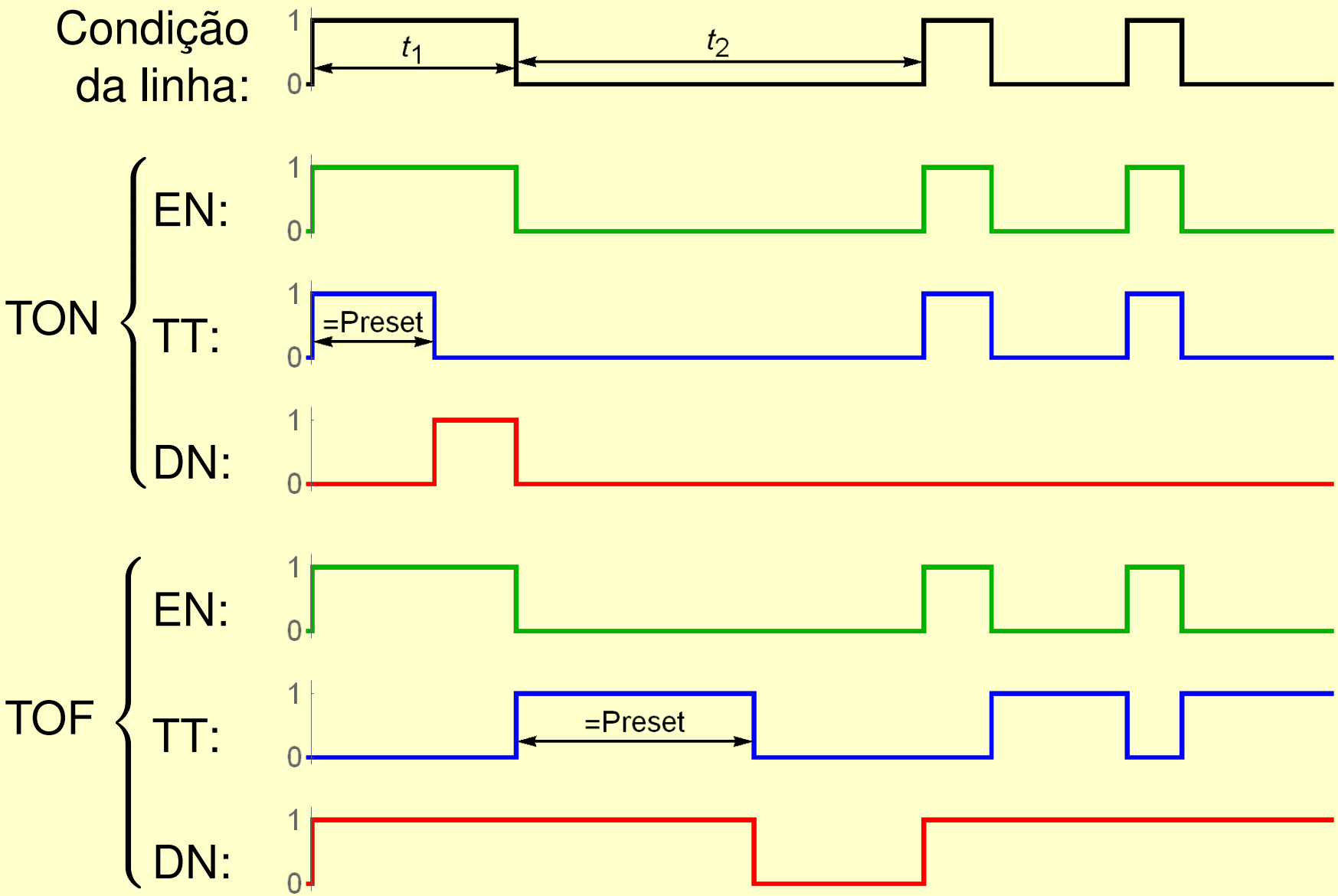


- **TON (Timer On-Delay), continuação**
- Quando a linha torna-se falsa (0), antes ou depois do tempo total ajustado, o temporizador é ressetado:
  - bits EN,TT e DN vão para 0;
  - ACC é zerado;
- Também é possível ressetar um TON com a instrução RES (Reset).

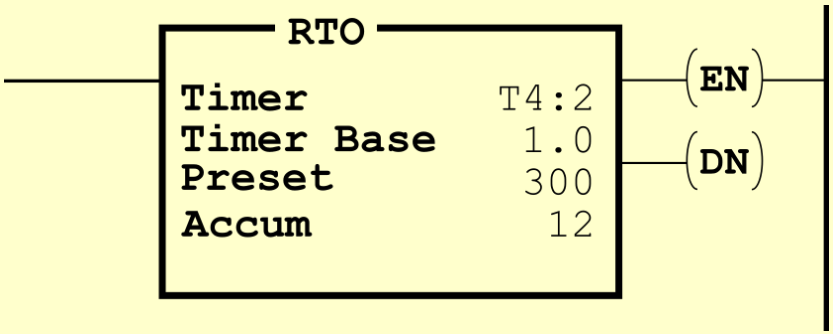


- **Instrução TOF (Timer Off-Delay)**
- Conta o tempo a partir do instante em que a condição da linha se torna falsa (atraso na desenergização);
- Da mesma forma que o TON, há a identificação exclusiva do timer (T4:n) e o ajuste de tempo no *Preset* e *Timer Base*;
- Bit 15 (EN=Enable): Linha habilitada = na condição lógica 1;
- Bit 14 (TT=Timer Timing): Temporização em andamento; linha está falsa mas ainda não atingiu tempo ajustado;
- Bit 13 (DN=Done): Se temporização já atingiu tempo ajustado, 1 $\rightarrow$ 0 ;
- Quando a quando a linha se torna verdadeira, TT $\rightarrow$ 0 e DN $\rightarrow$ 1;
- TOF não deve ser ressetado pela instrução RES.

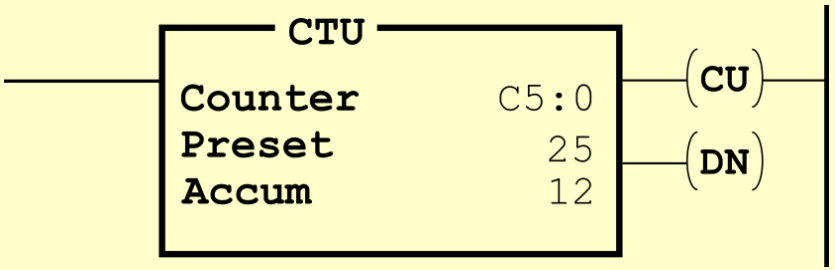



Condição da linha:
1
0
t_1
t_2
TON
EN:
TT:
DN:
=Preset
TOF
EN:
TT:
DN:
=Preset

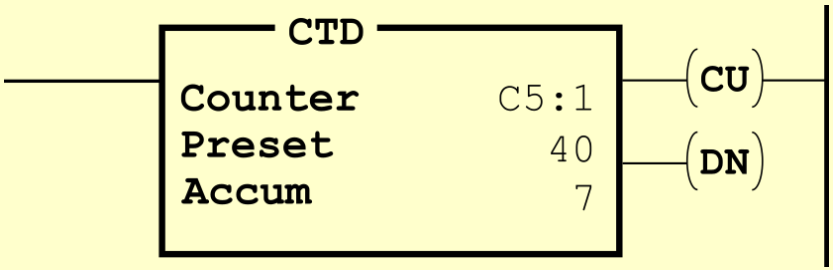
- **Instrução RTO (Retentive Timer On-Delay)**
- Similar ao TON, com os mesmos bits EN, TT e DN;
- Valor acumulado não é ressetado quando a linha se torna falsa, somente pela instrução RES.



- **Instrução CTU (Count Up)**
- Incrementa um contador a cada transição 0 $\rightarrow$ 1 na linha;
- Quando o valor acumulado (ACC) $>=$ valor ajustado (PRESET), o bit DN vai de 0 $\rightarrow$ 1;
- A condição lógica da linha é espelhada no bit CU (Count Up Enable);
- Contador também pode ser ressetado pela instrução RES.



- **Instrução CTD (Count Down)**
- Decrementa um contador a cada transição 0 $\rightarrow$ 1 na linha;
- Quando o valor acumulado (ACC) $>=$ valor ajustado (PRESET), o bit DN vai de 0 $\rightarrow$ 1;
- A condição lógica da linha é espelhada no bit CD (Count Down Enable);
- Contador também pode ser ressetado pela instrução RES.



Outras instruções

- END: marca de final do programa;
- Detecção de borda de subida ou descida, transformam um degrau em um pulso;
  - OSR = One Shot Rising, OSF=One Shot Falling
- Instruções matemáticas;
  - ADD, SUB, MUL, SQR, atribuição (MOV)
- Comparação, lógicas bit-a-bit
- Funções de controle;
- Consultar manuais de referência do produto.

OBRIGADO!

Este material foi desenvolvido para as diversas disciplinas da área de Eletrotécnica Geral da Escola Politécnica da Universidade de São Paulo pelos professores Milana Lima dos Santos e Eduardo Lorenzetti Pellini, sob a coordenação do professor Hernán Prieto Schmidt e supervisão do professor Giovanni Manassero Junior.

Agradecemos as valiosas contribuições do professor Lourenço Matakas Junior.

Referências:
Sergio Luiz Pereira. Controladores Lógicos Programáveis (Apostila), 2003.
Sergio Luiz Pereira, Lourenço Matakas Junior. CLP - Controladores Lógicos Programáveis; Laboratório de Automação.
Rockwell Automation, software RSLogix e manuais de referência