Capítulo 5

A programação Ladder

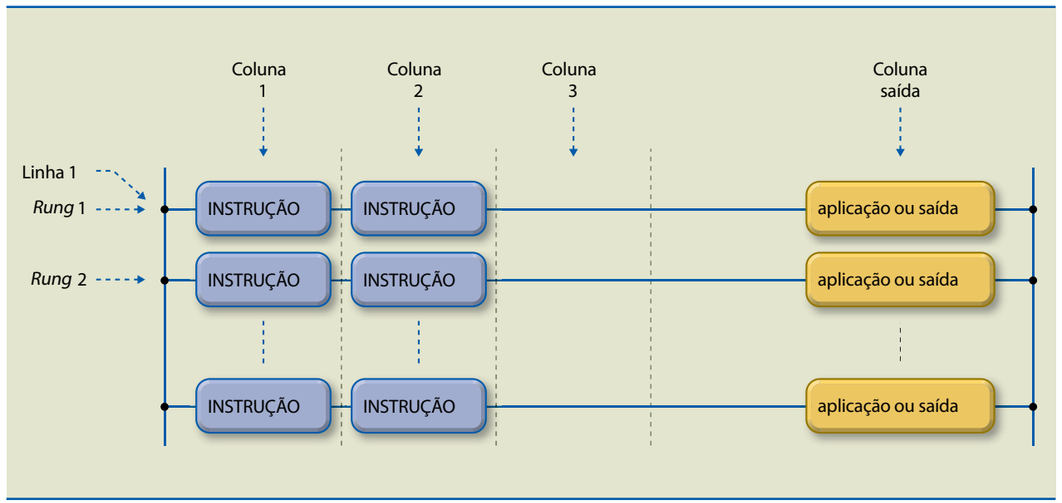
A Ladder foi uma das primeiras linguagens destinadas à programação de CLPs, criada para permitir que técnicos e engenheiros da área de automação com conhecimentos de lógica de relés e nenhum de programação conseguissem programar o CLP. Por esse motivo, ela se tornou a linguagem mais popular entre os programadores.

5.1 Diretivas básicas

As variáveis associadas aos elementos de entrada, saída, memória, temporizadores e contadores são denominadas operandos. O programa executa operações lógicas e aritméticas com esses operandos.

Na linguagem Ladder, as linhas de contatos (instruções) possuem a aparência de degraus (*rungs*) de uma escada (*ladder*), que podem ser associados a uma estrutura de colunas e linhas, conforme ilustra a figura 5.1. Em cada linha, as instruções correspondem ao programa, ou seja, ao processamento dos operandos, e o resultado é atribuído a outro operando no bloco "Saída", à direita.

Figura 5.1
Diagrama Ladder.



O número de linhas e colunas ou elementos e associações que cada *rung* admite varia conforme o fabricante do CLP e pode variar também de acordo com a UCP utilizada. Em geral, esses limites não representam preocupação ao usuário no desenvolvimento do programa de aplicação, pois, se o limite for ultrapassado, o *software* de programação apresentará uma mensagem de erro durante a compilação do programa.

Os operandos podem ser divididos em três classes:

- **Memória (M)** – Servem para o armazenamento dos resultados parciais, valores de constantes, dados de transmissão, valores de referência, receitas etc. Esses operandos podem ser livremente lidos e escritos pelo programa.
- **Entradas (I)** – Estão associados aos módulos de entrada. Podem ser lidos pelo programa, mas escritos apenas pelos módulos de entrada.
- **Saídas (Q)** – Estão associados aos módulos de saída. Podem ser livremente lidos e escritos pelo programa.

Os operandos, por sua vez, são divididos, inicialmente, em cinco tipos, segundo sua utilização e número de bits:

- **Bits (X)** – Utilizados para a implementação de lógica, ocupam 1 bit de memória.
- **Bytes (B)** – Utilizados para o armazenamento de caracteres ASCII, ocupam 8 bits.
- ***Words* (W)** – Utilizados para o armazenamento de valores numéricos inteiros, ocupam 16 bits.
- ***Double word* (D)** – Semelhante ao tipo W, ocupa 32 bits.
- ***Word long* (L) de 64 bits** – Semelhante ao tipo W, ocupa 64 bits.

Originalmente, na linguagem Ladder cada instrução correspondia aos contatos NA ou NF dos relés, cujo estado era definido pelo valor do operando (do tipo B) a ele associado. Na mesma época, as saídas eram as bobinas (operando tipo B). Com o tempo, os blocos de instruções passaram a contemplar contadores, temporizadores, operações aritméticas etc., que exigiram que fossem criados os tipos de operando citados anteriormente.

O objeto de estudo das instruções de programação Ladder será um CLP genérico, com os seguintes elementos:

- 8 entradas digitais.
- 8 saídas digitais.
- 4 entradas analógicas.
- 2 saídas analógicas.
- 6 saídas a relé.
- 1 entrada de contagem rápida de pulsos a 4 kHz.
- 32 temporizadores.
- 32 contadores.

5.2 Ligação das entradas e saídas digitais do CLP genérico

Essas instruções possuem apenas dois estados, que são representados pelos números binários “0” ou “1” e podem ser interpretados como desligado e ligado. Geralmente, estão associados a dispositivos elétricos como botoeiras, chaves liga e desliga, válvulas eletropneumáticas, pressostatos, termostatos etc.

5.2.1 Entradas digitais

As entradas digitais do CLP genérico são acionadas por botões, chaves, interfaces, *encoders* e outros dispositivos e podem ser do tipo NPN ou PNP. A entrada NPN é acionada com 0 V_{cc} (figura 5.2). Nessa configuração, o polo negativo é chaveado e o positivo permanece conectado. A entrada PNP é acionada com 24 V_{cc} (figura 5.3). Nessa configuração, o polo positivo é chaveado e o negativo permanece conectado.

Figura 5.2
Entrada digital NPN.

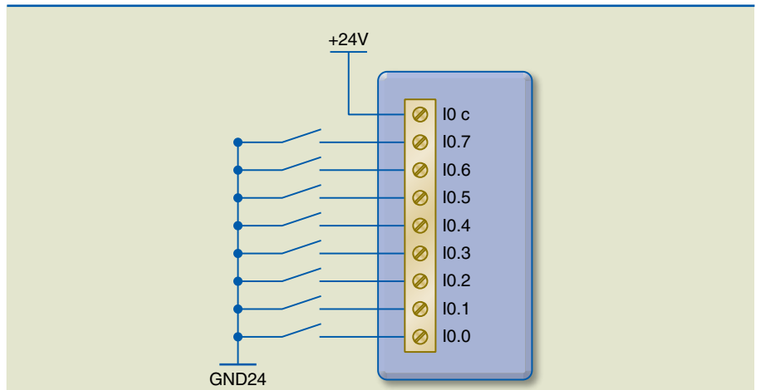
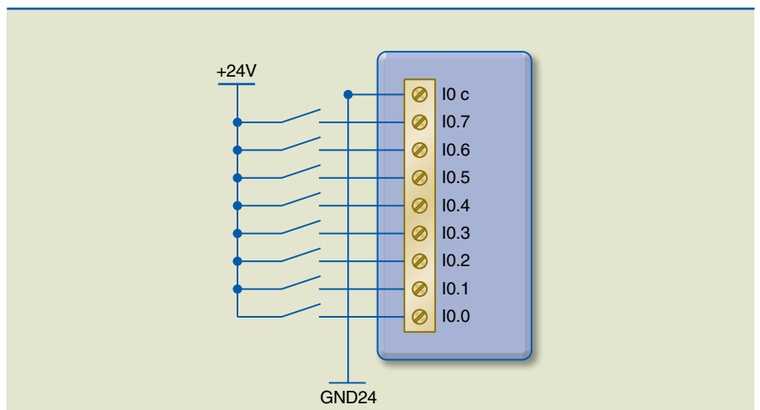


Figura 5.3
Entrada digital PNP.



Em nosso CLP genérico, utilizaremos entradas digitais PNP, que serão identificadas de I0.0 a I0.7. A identificação das entradas pode variar de acordo com o fabricante do CLP, usando, por exemplo, as letras “E” ou “X”. Da mesma forma, a identificação de qualquer componente também pode variar (saídas, temporizadores, contadores, memórias etc.).

5.2.2 Entrada rápida

A entrada rápida possui uma frequência máxima de 4 kHz e efetua contagem unidirecional. Em geral, a contagem é efetuada por borda de descida ou subida, dependendo exclusivamente do fabricante do CLP. Na figura 5.4 utiliza-se um *encoder* para demonstrar esse tipo de contagem. Essa entrada é acionada com 24 V_{cc} no borne de CR0+ e com GND no borne CR0–.

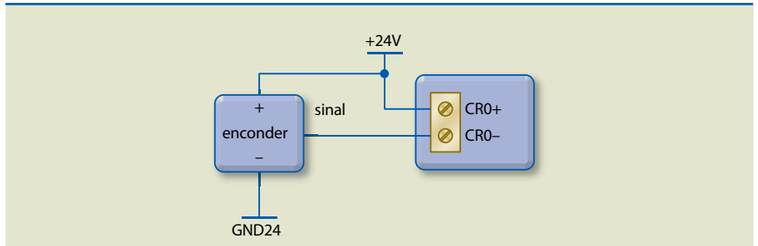


Figura 5.4
Entrada rápida.

5.2.3 Saídas digitais

De maneira análoga às entradas digitais, as saídas do CLP genérico também podem ser do tipo NPN ou PNP. Elas são acionadas de acordo com a programação feita pelo usuário. As saídas NPN, quando acionadas, fornecem para a carga o potencial GND (figura 5.5). As saídas PNP, quando acionadas, fornecem para a carga o potencial 24 V_{cc} (figura 5.6).

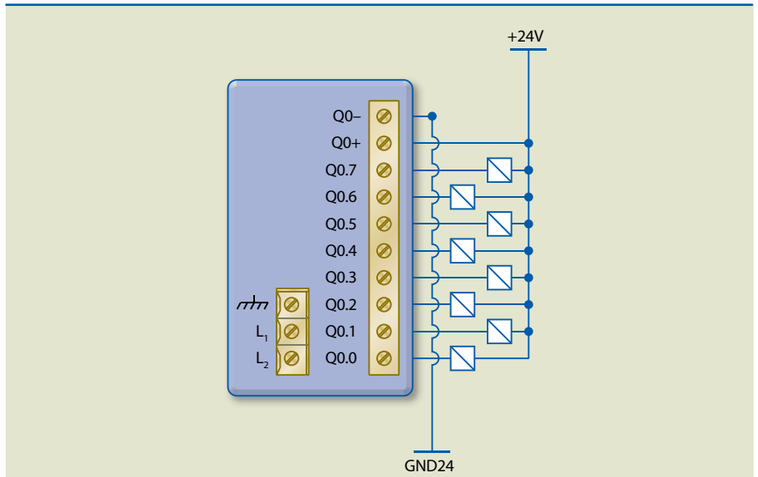
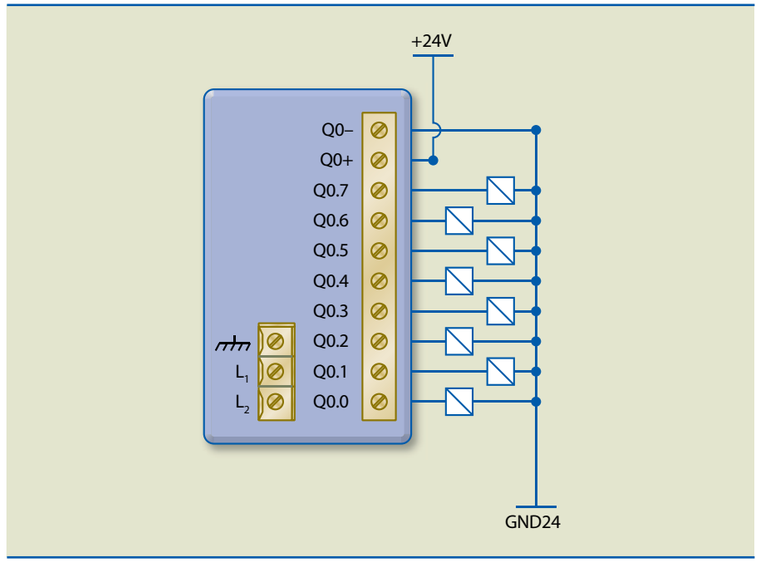


Figura 5.5
Saída digital NPN.

Figura 5.6
Saída digital PNP.

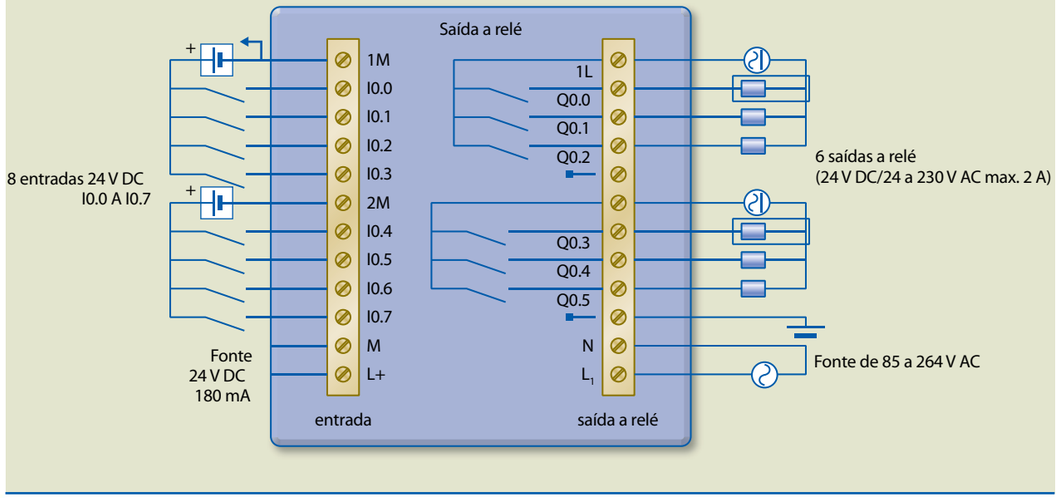


5.2.4 Saídas a relé

A saída a relé é muito utilizada por sua versatilidade em comandar cargas que trabalham com tensão alternada ou contínua. Quando ativada, a saída chaveia um contato de relé.

A figura 5.7 mostra saídas a relé para cargas em corrente alternada ou contínua. Nela, pode-se observar que a polaridade na ligação é indiferente, porém, deve-se ficar atento aos limites de corrente e de tensão do contato. Consulte essas informações no manual do CLP.

Figura 5.7
Saídas a relé para cargas em CC ou CA.



5.3 Contatos NA/NF

O diagrama de contatos Ladder funciona como um esquema elétrico cujos principais elementos são o contato normalmente aberto, o contato normalmente fechado e a bobina do relé.

5.3.1 Contato NA

Essa instrução funciona do seguinte modo: quando o bit associado a um contato normalmente aberto for acionado, o contato fechará; caso contrário, ele permanecerá aberto. Outra maneira de entender é imaginando um botão com o contato normalmente aberto: enquanto esse botão estiver solto, o contato ficará aberto, porém, ao ser pressionado, o contato do botão fechará.

Caso o botão NA esteja em um circuito elétrico, ocorrerá a passagem de corrente elétrica nos componentes do circuito. Se houver uma carga em série com esse botão e uma tensão de alimentação, a carga será acionada.

A figura 5.8 mostra o circuito elétrico, e a figura 5.9, a representação gráfica de um contato NA em diagrama Ladder. Note que, na figura 5.8, em cima da instrução NA, aparece o endereço do operando relacionado a ela.

Figura 5.8
Circuito elétrico: contato NA (botão 1).

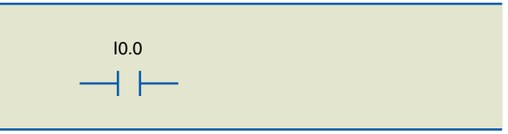


Figura 5.9
Representação gráfica do contato NA em diagrama Ladder.

Na figura 5.9, o contato NA relacionado ao operando I0.0 (entrada) estará aberto se a entrada estiver desacionada (nível lógico "0") e fechado se a entrada estiver acionada (nível lógico "1").

5.3.2 Contato NF

Essa instrução funciona do seguinte modo: quando o bit associado a um contato normalmente fechado for acionado, o contato abrirá; caso contrário, ele permanecerá fechado. Outra maneira de entender é imaginando um botão com o contato normalmente fechado: enquanto esse botão estiver solto, o contato ficará fechado, porém, ao ser pressionado, o contato do botão abrirá.

Caso o botão NF esteja em um circuito elétrico, não ocorrerá passagem de corrente elétrica. Se houver uma carga em série com esse botão e uma tensão de alimentação, a carga será desligada.

A figura 5.10 mostra o circuito elétrico, e a figura 5.11, a representação gráfica de um contato NF em diagrama Ladder. Note que, na figura 5.10, em cima da instrução NF, aparece o endereço do operando relacionado a ela.

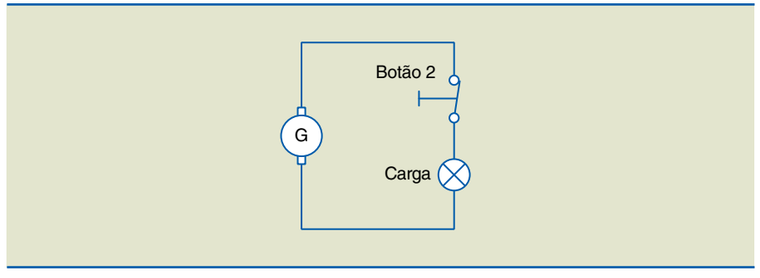


Figura 5.10
Circuito elétrico: contato NF (botão 2).

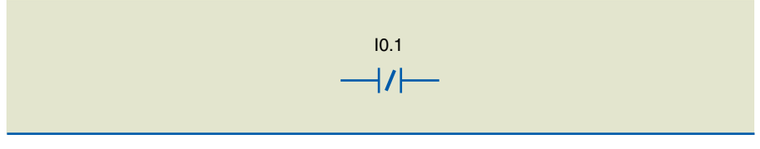


Figura 5.11
Representação gráfica do contato NF em diagrama Ladder.

Na figura 5.11, o contato NF relacionado ao operando I0.1 (entrada) estará fechado se a entrada estiver desligada (nível lógico "0") e aberto se a entrada estiver acionada (nível lógico "1").

5.4 Saída simples/saída complementar

A saída tem por base a ideia de continuidade lógica a ser garantida entre os extremos das linhas de programação. Uma saída será verdadeira se todas as instruções declaradas na linha lógica forem verdadeiras.

5.4.1 Saída simples

Essa instrução, ao ser acionada, transfere para o endereço associado a ela o valor da tensão que estiver em sua entrada. Por exemplo, em circuitos elétricos, utilizam-se diretamente relés ou contatores para acionar cargas como motores, resistências etc. Na figura 5.12, quando acionam-se o botão 1, energizam-se a bobina do relé 1, o que, consequentemente, fecha os contatos 13 e 14, acionando a carga.

Em nosso CLP genérico, o relé 1 representa uma saída simples que tem como operando o endereço de saída Q0.0. A figura 5.13 mostra a representação gráfica de uma saída simples. Note que, em cima da instrução bobina, aparece o endereço do operando relacionado a ela.

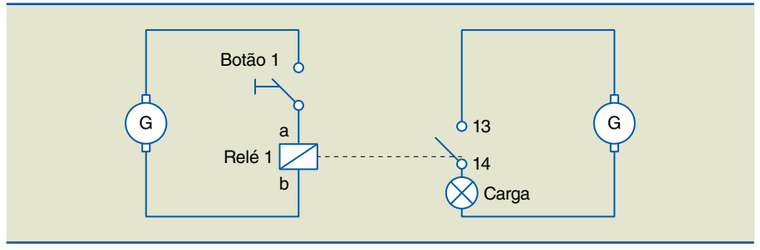


Figura 5.12
Circuito elétrico utilizando relé para acionamento de uma carga.

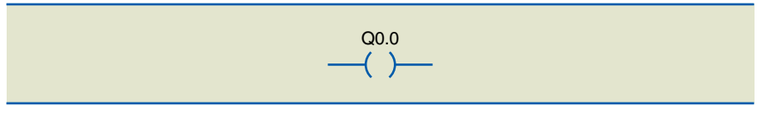


Figura 5.13
Representação gráfica de uma saída simples em diagrama Ladder.

5.4.2 Saída complementar

Essa instrução, ao ser acionada, transfere para o endereço associado a ela o valor de tensão oposto ao de sua entrada. Na figura 5.14, quando acionamos o botão 1, energizamos a bobina do relé 1, o que, consequentemente, abre os contatos 21 e 22, desligando a carga. Nesse caso, o relé 1 representa uma saída complementar que tem como operando o endereço de saída Q0.1. A figura 5.15 mostra a representação gráfica de uma saída complementar. Note que, em cima da instrução bobina, aparece o endereço do operando relacionado a ela.

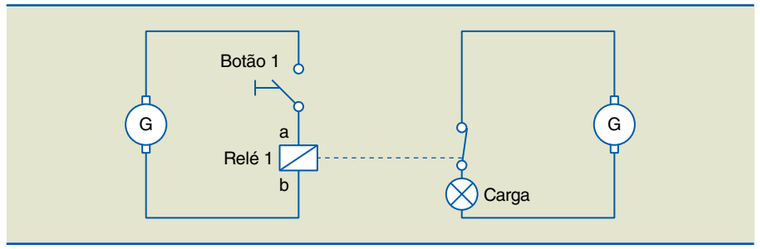


Figura 5.14
Circuito elétrico utilizando relé para desacionar uma carga.

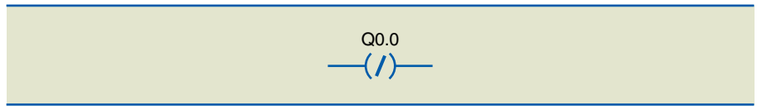


Figura 5.15
Representação gráfica de uma saída complementar em diagrama Ladder.

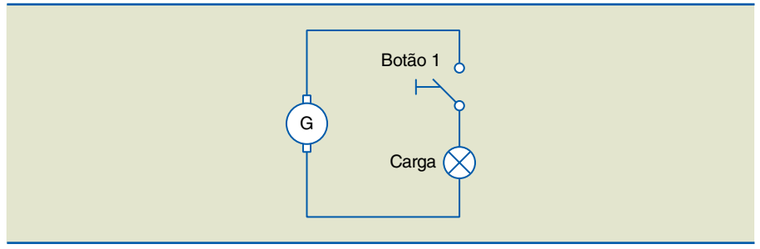
Agora que já conhecemos algumas instruções e suas representações gráficas, vamos ver alguns exemplos de programas, inicialmente utilizando circuitos simples, para facilitar a compreensão.

Exemplos

1. Funcionamento da instrução NA e da saída simples.

Faça o diagrama Ladder para o circuito da figura 5.16.

Figura 5.16
Circuito elétrico – contato NA representado pelo botão 1.



Solução:

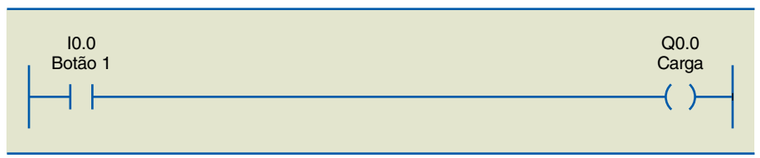
O programa começa com a identificação das entradas e das saídas. Faça uma tabela mostrando cada um desses endereços e relacione-os a uma simbologia que identifique a função das instruções (tabela 5.1). Em programas complexos, isso é essencial na resolução de problemas e em modificações técnicas. Se possível, adicione um comentário.

Tabela 5.1
Endereços, símbolos e comentários

Endereço	Símbolo	Comentário
I0.0	Botão 1	Aciona a carga
Q0.0	Carga	Pode ser motor, lâmpada, relé, resistência etc.

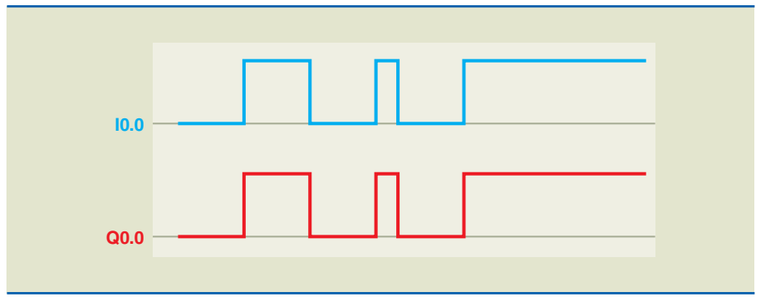
O programa em Ladder para o circuito, apresentado na figura 5.17, mostra que a saída simples (Q0.0) será acionada somente quando a entrada (I0.0) for acionada, ou seja, quando estiver em nível lógico "1".

Figura 5.17
Programa em Ladder para o circuito da figura 5.16.



A figura 5.18 ilustra o diagrama de estado da entrada (I0.0) e da saída (Q0.0) em função do tempo. Note que o estado da saída acompanha o estado da entrada.

Figura 5.18
Diagrama de estado da entrada NA (I0.0) e da saída simples (Q0.0).



O esquema de ligação no CLP genérico está representado na figura 5.19, que mostra somente as entradas e saídas digitais.

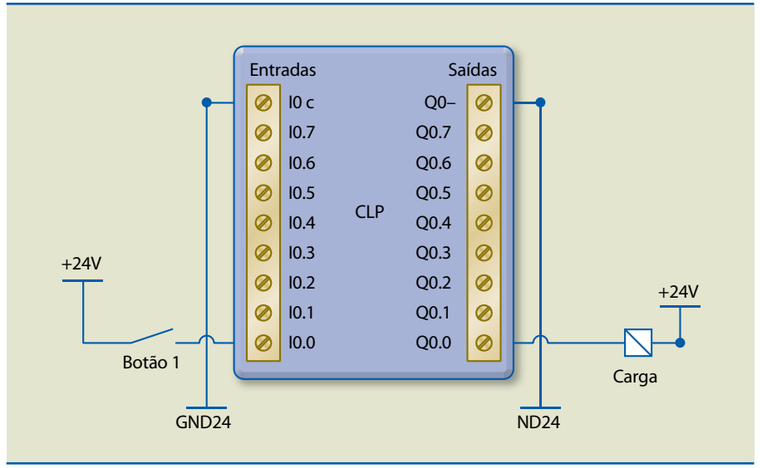


Figura 5.19
Esquema de ligação no CLP genérico.

2. Funcionamento da instrução NF e da saída simples.

Faça o diagrama Ladder para o circuito da figura 5.20.

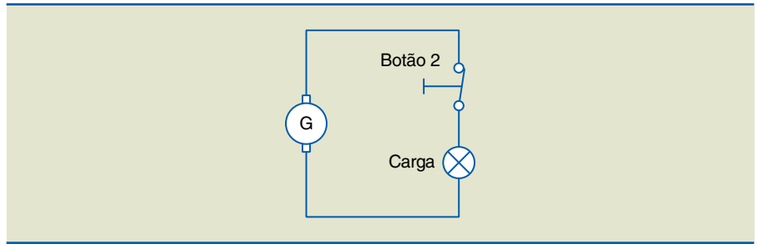


Figura 5.20
Circuito elétrico – contato NF representado pelo botão 2.

Solução:

Inicie a programação pela tabela de endereços, símbolos e comentários (tabela 5.2). Quanto mais informações forem incluídas no diagrama, mais fácil será modificá-lo caso necessário.

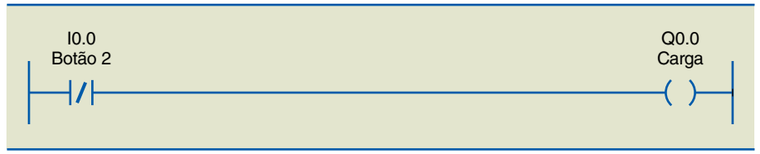
Tabela 5.2
Endereços, símbolos e comentários

Endereço	Símbolo	Comentário
I0.0	Botão 2	Desacionar a carga quando pressionado
Q0.0	Carga	Pode ser motor, lâmpada, relé, resistência etc.

O programa em Ladder para o circuito, apresentado na figura 5.21, mostra que a saída simples (Q0.0) será desacionada (nível lógico "0") somente quando a entra-

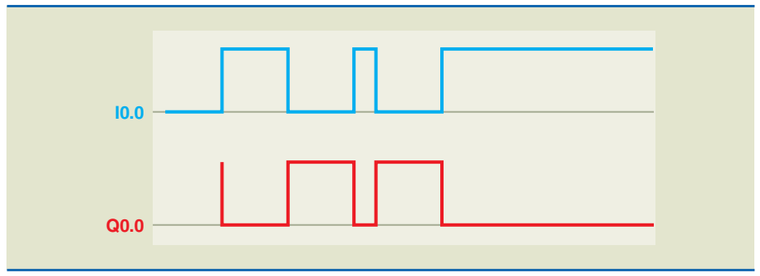
da (I0.0) for acionada, ou seja, quando estiver em nível lógico “1”. Portanto, a saída é o oposto da entrada.

Figura 5.21
Programa em Ladder para o circuito da figura 5.20.



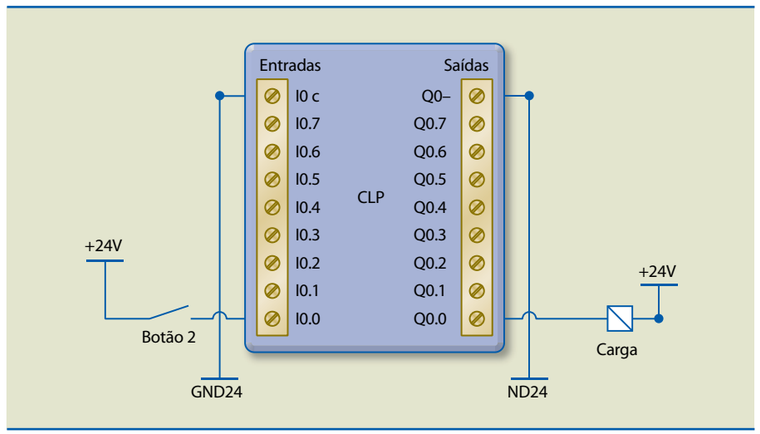
A figura 5.22 ilustra o diagrama de estado da entrada (I0.0) e da saída (Q0.0) em função do tempo. Note que o estado da saída é o contrário do estado da entrada.

Figura 5.22
Diagrama de estado da entrada NF (I0.0) e da saída simples (Q0.0).



O esquema de ligação no CLP está representado na figura 5.23. Observe que a ligação não foi alterada: tanto a entrada como a saída permanecem da mesma forma; porém, o funcionamento é diferente. Essa é uma característica dos CLPs. Sem alteração na ligação elétrica, mas com mudança na programação, altera-se o funcionamento do sistema.

Figura 5.23
Esquema de ligação no CLP genérico.



3. Funcionamento da instrução NA e da saída complementar.

Faça o diagrama Ladder para o circuito da figura 5.24.

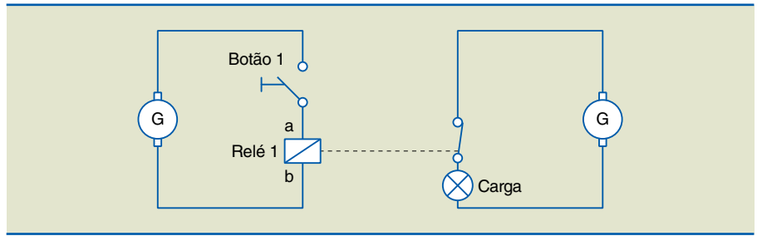


Figura 5.24
Circuito elétrico utilizando relé para desacionar uma carga.

Solução:

Inicie a programação pela tabela de endereços, símbolos e comentários (tabela 5.3). Quanto mais informações forem incluídas no diagrama, mais fácil será modificá-lo caso necessário.

Tabela 5.3
Endereços, símbolos e comentários

Endereço	Símbolo	Comentário
I0.0	Botão 1	Desacionar a carga quando pressionado
Q0.0	Carga	Pode ser motor, lâmpada, relé, resistência etc.

O programa em Ladder para o circuito, apresentado na figura 5.25, mostra que, quando acionada, a entrada (I0.0) transferirá para a saída complementar (Q0.0) o inverso do sinal da entrada associada a ela (nível lógico “0”). Isto significa que somente quando a entrada (I0.0) for acionada, ou seja, quando estiver em nível lógico “1”, a saída será o oposto da entrada (nível lógico “0”).

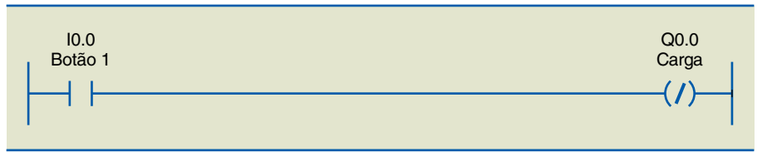


Figura 5.25
Programa em Ladder para o circuito da figura 5.24.

A figura 5.26 ilustra o diagrama de estado da entrada (I0.0) e da saída complementar (Q0.0) em função do tempo. Note que o estado da saída é o contrário do estado da entrada.

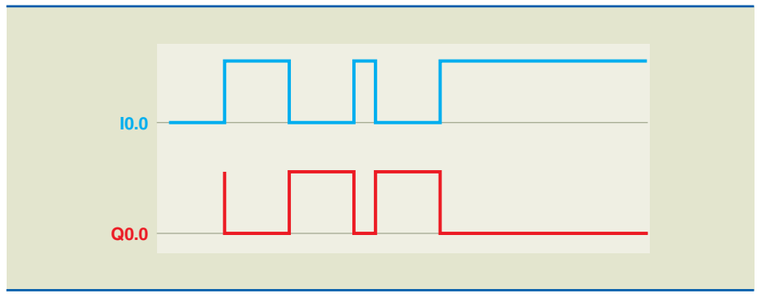
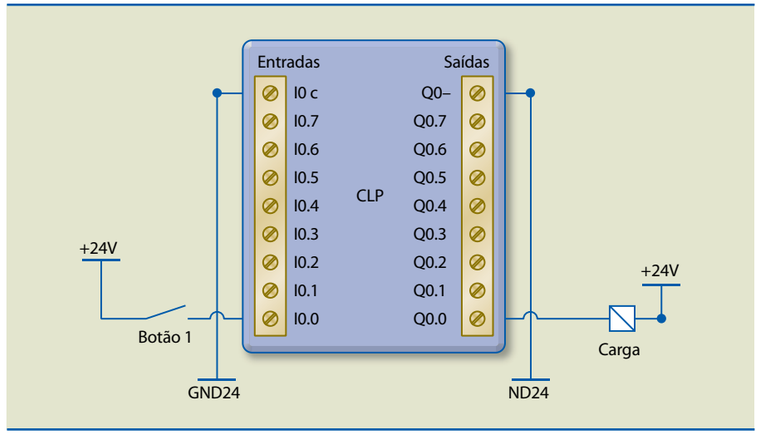


Figura 5.26
Diagrama de estado da entrada NA (I0.0) e da saída complementar (Q0.0).

O esquema de ligação no CLP está representado na figura 5.27, mostrando de novo que, com o funcionamento diferente, pode-se utilizar a mesma ligação externa.

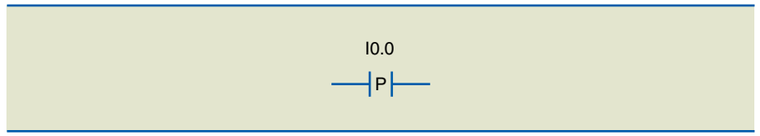
Figura 5.27
Esquema de ligação no CLP genérico.



5.5 Contato por borda positiva

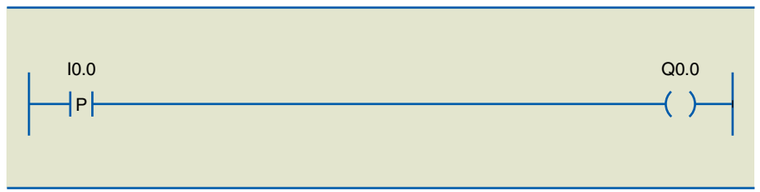
Outra das instruções consideradas especiais, por possuírem características e aplicações peculiares, é o contato por borda positiva (figura 5.28). Essa instrução gera um pulso na saída associada a ela. O pulso tem o período de 1 *scan* e inicia-se quando a entrada faz a passagem do nível lógico "0" para o "1".

Figura 5.28
Representação gráfica do contato por borda positiva.



Um programa utilizando esse tipo de instrução pode ser visto na figura 5.29.

Figura 5.29
Lógica utilizando contato por borda positiva e saída simples.



O diagrama de estado da entrada por borda positiva (I0.0) e da saída simples (Q0.0), apresentado na figura 5.30, demonstra a aplicação dessa instrução. Observe que, quando a entrada (I0.0) é acionada, ou seja, na passagem do nível lógico "0" para o "1", na saída associada a essa entrada aparece um pulso com duração de 1 *scan*. Na descida, na passagem do nível lógico "1" para o "0", nada acontece à saída.

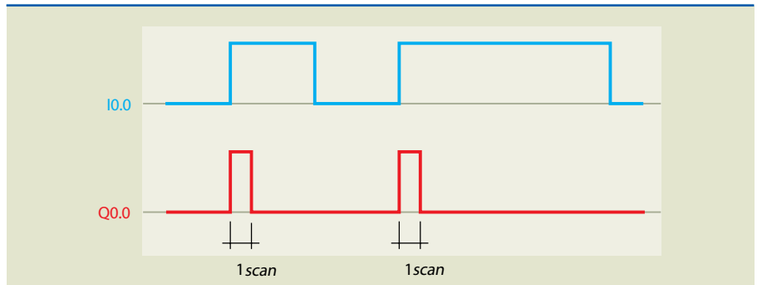


Figura 5.30
Diagrama de estado da entrada por borda positiva e da saída simples.

5.6 Contato por borda negativa

A instrução contato por borda negativa (figura 5.31) gera um pulso na saída associada a ela. Esse pulso tem o período de 1 *scan* e inicia-se quando a entrada faz a passagem do nível lógico "1" para o "0".

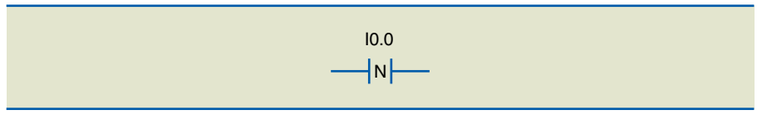


Figura 5.31
Representação gráfica do contato por borda negativa.

Um programa utilizando esse tipo de instrução pode ser visto na figura 5.32.

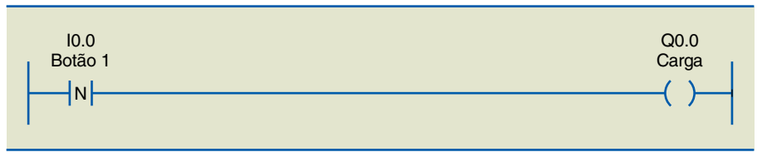


Figura 5.32
Lógica utilizando contato por borda negativa e saída simples.

O diagrama de estado da entrada por borda negativa (I0.0) e da saída simples (Q0.0), apresentado na figura 5.33, demonstra a aplicação dessa instrução. Observe que, quando a entrada (I0.0) é acionada, nada acontece; porém, ao ser desligada, na passagem do nível lógico "1" para o "0", na saída associada a essa entrada aparece um pulso com duração de 1 *scan*.

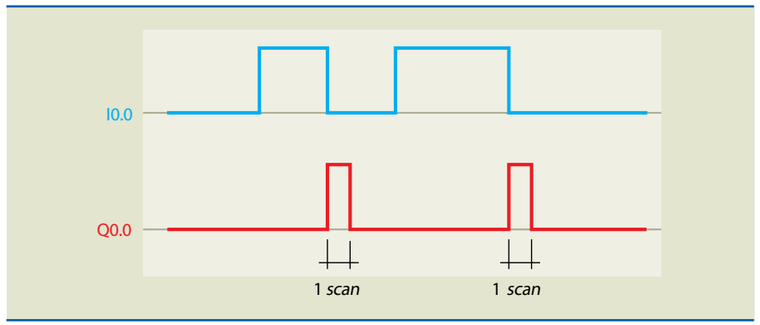


Figura 5.33
Diagrama de estado da entrada por borda negativa e da saída simples.

5.7 Saída *set* e saída *reset*

As instruções *set* e *reset* são utilizadas para memorização dos sinais de saída do CLP.

A instrução *set* (figura 5.34) serve para acionar e manter acionado um operando de saída quando, na entrada associada a ela, houver um pulso (passagem do nível lógico "0" para o "1"). Mesmo que a entrada associada à instrução *set* passe para o nível lógico "0" (transição do nível lógico "1" para o "0"), a saída permanecerá acionada.

A instrução *reset* (figura 5.35) serve para desacionar e manter desacionado um operando de saída quando, na entrada associada a ela, houver um pulso (passagem do nível lógico "0" para o "1"). A instrução *reset* permanecerá em "0" mesmo que a entrada associada a ela passe para o nível lógico "0" (transição do nível lógico "1" para o "0").

Figura 5.34
Representação gráfica saída *set*.

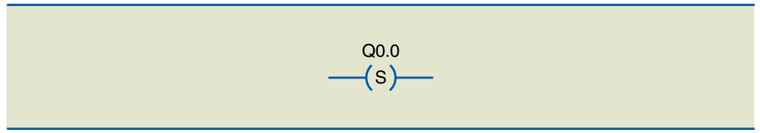
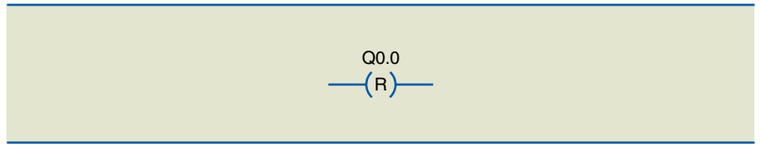
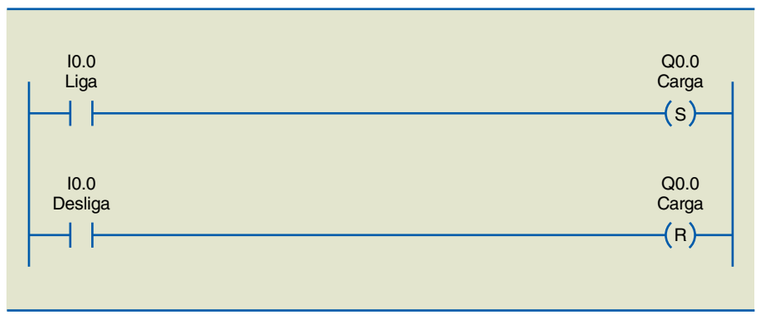


Figura 5.35
Representação gráfica saída *reset*.



Em resumo, a saída *set* liga um operando quando a entrada associada a ele passa do nível lógico "0" para o "1", e a saída *reset* desliga o operando quando a entrada associada a ele passa de "0" para "1" (figura 5.36).

Figura 5.36
Lógica Ladder utilizando as saídas *set* e *reset*.



Analisando o diagrama de estado do programa da figura 5.36, pode-se notar que a saída *set*, carga (Q0.0), é acionada quando a entrada liga (I0.0) passa do nível lógico "0" para o "1" e permanece acionada mesmo quando a entrada liga (I0.0) passa para o nível lógico "0". A saída carga (Q0.0) só será desligada quando a entrada desliga (I0.1) for acionada (figura 5.37).

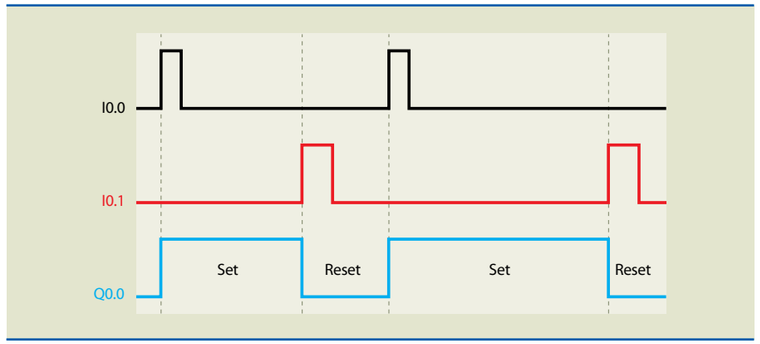


Figura 5.37
Saídas *set* e *reset*.

5.8 Memória ou *flag*

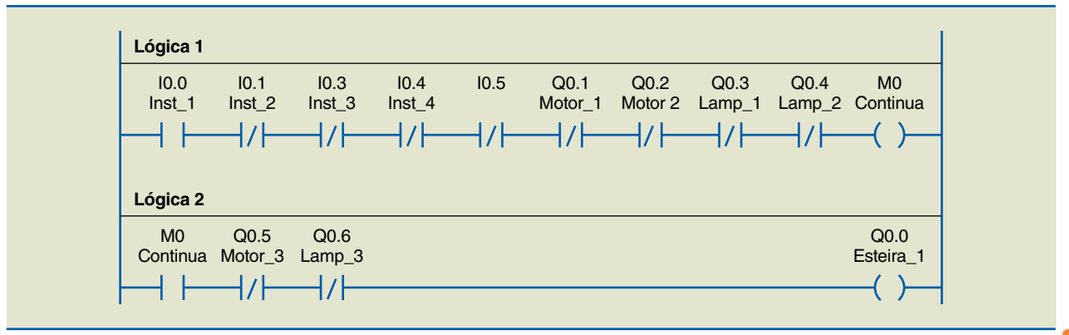
Essas instruções podem ser utilizadas como entradas ou saídas. Elas são bits de memória interna que, em nosso caso, podem ser endereçadas conforme a tabela 5.4.

Tabela 5.4
Endereços internos

Endereços	Tamanho	Descrição
M0 a M255	1 bit (nível "1" ou nível "0")	Bits de memória
MR0 a MR255	1 bit (nível "1" ou nível "0")	Bits retentivos
MB0 a MB255	1 byte (de 0 a 255)	Bytes de memória
MBR0 a MBR255	1 byte (de 0 a 255)	Bytes retentivos
MW0 a MW255	2 bytes (de 0 a 65 535)	*Word* de memória
MWR0 a MWR255	2 bytes (de 0 a 65 535)	*Word* retentivo

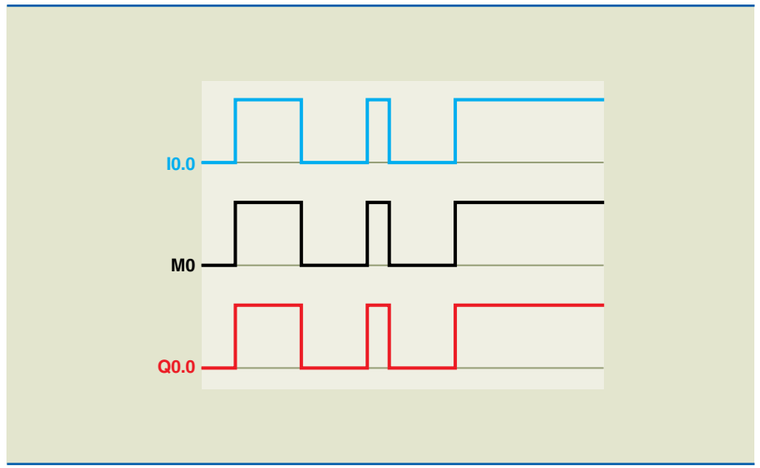
A figura 5.38 mostra que as instruções não couberam na linha lógica 1. Nesse caso, foi necessária a utilização de um bit de memória (M0) como saída. Note que a representação gráfica é a mesma de uma saída simples. Na linha lógica 2, para dar sequência, utiliza-se o endereço de instrução (M0) como entrada. Observe que a representação gráfica é a mesma de uma entrada. Esse recurso economiza saída física do CLP e permite o uso de linhas longas, com muitas instruções.

Figura 5.38
Utilização de memória para sequência lógica.



O diagrama de estado da figura 5.39 demonstra a versatilidade da utilização de memória em linhas lógicas complexas. Quando a linha lógica 1 for verdadeira, a saída (M0) também será e atuará na lógica 2, que, se também for verdadeira, acionará a saída (Q0.0).

Figura 5.39
Diagrama de estado utilizando memória.



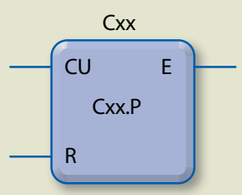
5.9 Contadores

Os contadores são usados quando se deseja contar o número de vezes que determinado evento ocorre – por exemplo, peças produzidas, operações realizadas etc.

5.9.1 Contador UP

O contador UP, denominado contador crescente (figura 5.40), incrementa uma unidade toda vez que o contato associado à entrada (CU) passa do estado lógico "0" para o "1", até atingir o valor predeterminado (valor do *preset*). Quando o valor atual atingir o valor do *preset*, a saída (E) do contador será acionada, passando do nível lógico "0" para o "1". Quando a entrada associada ao *reset* (R) do contador for acionada, passando do nível lógico "0" para o "1", o valor atual do contador será zerado, podendo reiniciar a contagem assim que a entrada associada ao *reset* voltar ao estado inicial (nível lógico "0").

Figura 5.40
Símbolo gráfico do contador UP.



Observe na tabela 5.5 os parâmetros do contador UP.

Tabela 5.5
Parâmetros do contador UP

C C CU E R	Cxx: em que xx é o número do contador, de 0 a 31, definido pelo usuário.
C01 CU E PRESET: = R	C01.P: valor do *preset*, definido pelo usuário. É um número inteiro na faixa de 0 a 65 535. C01.V: valor atual da contagem, definido por *software*. Utilizado para mostrar o número atual da contagem em uma IHM (interface homem-máquina), é incrementado por pulso aplicado na entrada (CU) do contador. É um número inteiro que varia de 0 a 65 535.
C01 CU E R	CU: entrada do contador, definida pelo usuário. Recebe bit ("0" ou "1") da entrada associada a ela.
C01 CU E R	R: *reset*, definido pelo usuário. É ativado pela entrada associada a ele. Quando acionado, zera o valor atual da contagem, reiniciando-a assim que a entrada volta ao nível lógico "0". Recebe bit ("0" ou "1") da entrada associada a ele.
C01 CU E R	E: saída do contador, definida por *software*. Quando o valor atual da contagem se iguala ao valor do *preset*, ela é ativada. Coloca nível lógico "1" no operando associado a ela.

Exemplo

Um exemplo de aplicação do contador UP é a limitação da quantidade de peças produzidas por uma máquina. Quando forem produzidas seis peças, o processo deve ser interrompido para sua retirada. Depois disso, deve ser reiniciado.

Solução:

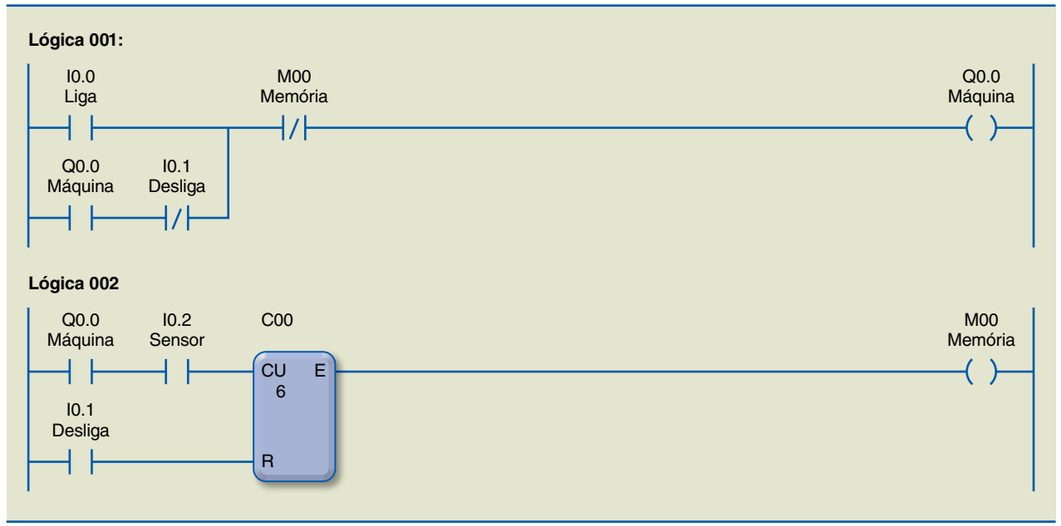
Os materiais necessários são: dois botões de contato momentâneo com retorno por mola, um sensor indutivo e o CLP genérico. Os endereços, símbolos e comentários são apresentados na tabela 5.6.

Tabela 5.6
Endereços, símbolos e comentários

Endereço	Símbolo	Comentário
I0.0	Liga	Botão de contato momentâneo, retorno por mola.
I0.1	Desliga	Botão de contato momentâneo, retorno por mola.
I0.2	Sensor	Sensor indutivo colocado na entrada do contador UP.
Q0.0	Máquina	Saída do CLP aciona a máquina.
M0	Bit de memória	Quando acionado, desliga a máquina.

Figura 5.41
Diagrama Ladder.

A figura 5.41 mostra o programa em diagrama Ladder.



No exemplo, alguns importantes conceitos de comandos elétricos são utilizados na primeira linha (*rung*). A memória (M00) está associada em série com o endereço liga (I0.0). Essa linha só será verdadeira se a memória (M00) não estiver acionada, estabelecendo uma dependência de funcionamento, ou seja, a máquina (Q0.0) somente será acionada se a memória (M00) estiver desacionada.

Ainda na lógica 1, a segunda linha utiliza o conceito de selo elétrico executado pelo endereço de saída da máquina (Q0.0). Essa lógica depende apenas do endereço liga (I0.0) para acionar a saída, que se mantém fechada por meio do endereço de saída da máquina (Q0.0) realocado como entrada. O desligamento da máquina depende do endereço desliga (I0.1) ou do endereço de memória (M00).

Na lógica 2, o contador foi parametrizado para contar até 6. Quando o valor atual atinge o valor do *preset*, a saída (E) do contador vai para nível lógico "1", acionando a memória (M00), também com nível lógico "1". Ao ser acionada, desliga o selo da lógica 1.

Os pulsos para a contagem são fornecidos pelo sensor (I0.2), mas somente são válidos quando a entrada de máquina (Q0.0) está acionada, ou seja, no nível lógico "1". Quando a memória (M00) é acionada, o sistema fica paralisado, podendo ser reiniciado ao acionar o operando desliga (I0.1), que resseta o contador e prepara a lógica "1" para ser acionada. O operando liga (I0.0) reinicia o processo. A figura 5.42 mostra o diagrama de estado.

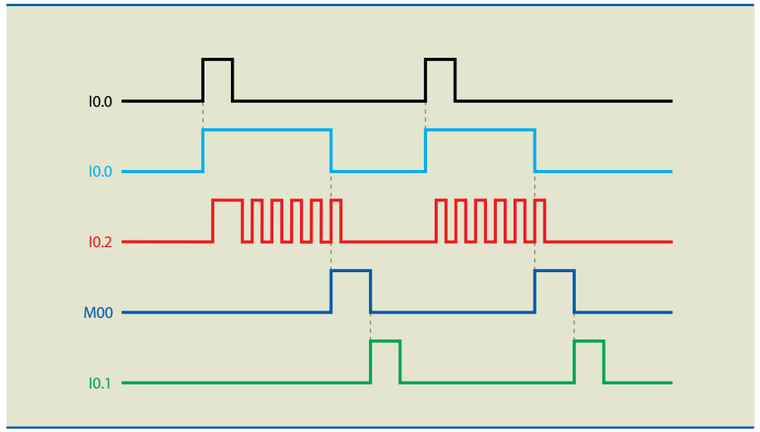


Figura 5.42
Diagrama de estado.

5.9.2 Contador DOWN

Nesse tipo de contador, o valor atual (Cxx.V) é carregado com o valor do *preset* (Cxx.P). O contador DOWN, denominado contador decrescente (figura 5.43), decrementa uma unidade toda vez que o contato associado à entrada (CD) passa do estado lógico "0" para o "1". Quando o valor atual do contador chegar a zero, a saída (E) do contador será acionada, passando do nível lógico "0" para o "1". Quando a entrada associada ao *preset* (P) do contador for acionada, passando do nível lógico "0" para o "1", o valor atual do contador será carregado com o valor do *preset* e sua saída voltará ao estado inicial (nível lógico "0"), podendo reiniciar a contagem.

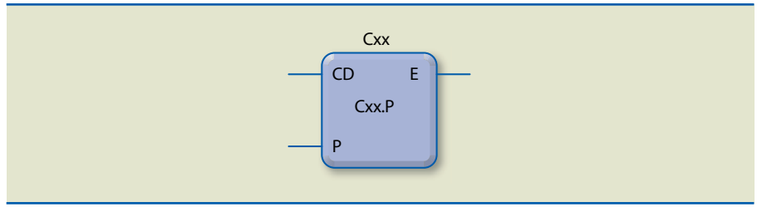


Figura 5.43
Símbolo gráfico do contador DOWN.

Tabela 5.7
Parâmetros do contador DOWN

A tabela 5.7 apresenta os parâmetros do contador DOWN.

C C CD E P	Cxx: em que xx é o número do contador, de 0 a 31, definido pelo usuário.
C01 CD E PRESET: = P	Cxx.P: valor do *preset*, definido pelo usuário. É um número inteiro na faixa de 0 a 65 535. Cxx.V: valor atual da contagem, definido por *software*. Utilizado para mostrar o número atual da contagem em uma IHM (interface homem-máquina), é decrementado por pulso aplicado na entrada (CD) do contador. É um número inteiro que varia de 0 a 65 535.
C01 CD E P	CD: entrada do contador, definida pelo usuário. Recebe bit (“0” ou “1”) da entrada associada a ela.
C01 CD E P	P: *preset*, definido pelo usuário. É ativado pela entrada associada a ele. Quando acionado, carrega com o valor do *preset* o valor atual. Recebe bit (“0” ou “1”) da entrada associada a ele.
C01 CD E P	E: saída do contador, definida por *software*. Quando o valor atual da contagem se iguala a zero, ela é ativada. Coloca bit (“0” ou “1”) no operando associado a ela.

Exemplo

Em um sistema produtivo, foi implantado um dispositivo que retira peças de uma esteira e as coloca em um recipiente que comporta seis peças. Quando o recipiente estiver cheio, o sistema deve parar a esteira e sinalizar ao operador para a retirada do recipiente com as peças produzidas. Quando o operador colocar um recipiente vazio na esteira, o sistema deve reiniciar.

Solução:

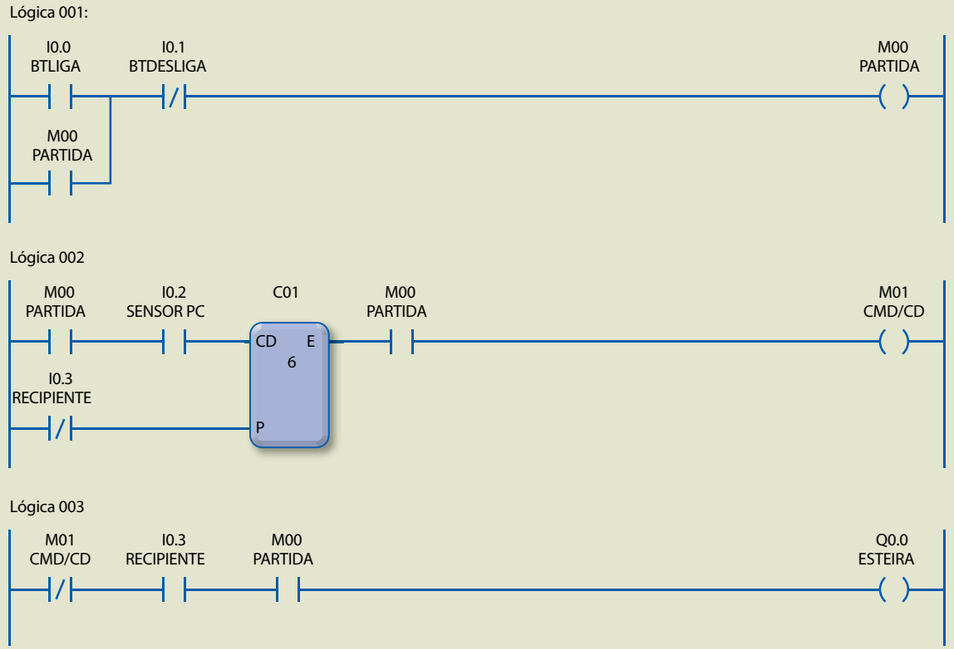
Os materiais necessários são: dois botões de contato momentâneo com retorno por mola, um sensor capacitivo, uma chave fim de curso e o CLP genérico. Os endereços, símbolos e comentários são apresentados na tabela 5.8.

Tabela 5.8
Endereços, símbolos e comentários

Endereço	Símbolo	Comentário
I0.0	BTLiga	Botão de contato momentâneo, retorno por mola.
I0.1	BTDesliga	Botão de contato momentâneo, retorno por mola.
I0.2	Sensor	Sensor capacitivo colocado na entrada do contador DOWN.
I0.3	Chave	Chave fim de curso para detectar presença de recipiente.
M00	Partida	Bit de memória inicia o processo.
M01	CMD/CD	Bit de memória da saída do contador bloqueia/libera a esteira.
Q0.0	Esteira	Acionamento da esteira.

A figura 5.44 mostra o programa em diagrama Ladder.

Figura 5.44
Diagrama Ladder.



O programa para esse exemplo foi feito com três lógicas, porém existem outras maneiras de executar a mesma tarefa. Essas lógicas também ilustram conceitos da programação Ladder.

A lógica 1 serve para ligar e desligar o sistema. O conceito de selo elétrico surge de novo e a memória é utilizada como recurso para economizar saída.

A lógica 2 apresenta o funcionamento do contador DOWN. O endereço *start* (M00) aparece como bloqueio, para que não haja contagem sem o sistema estar ligado. O endereço recipiente (I0.3), quando acionado, atualiza o valor do contador, preparando-o para uma nova contagem. O endereço CMD/CD (M01) atua diretamente sobre o funcionamento da esteira. Quando acionado, desliga a esteira e, quando desacionado, prepara o sistema para ser ligado.

A lógica 3 tem o funcionamento dependente de três endereços: (M01), (I0.3) e (M00). Quando os três são verdadeiros, a esteira (Q0.0) é acionada.

5.9.3 Contador rápido

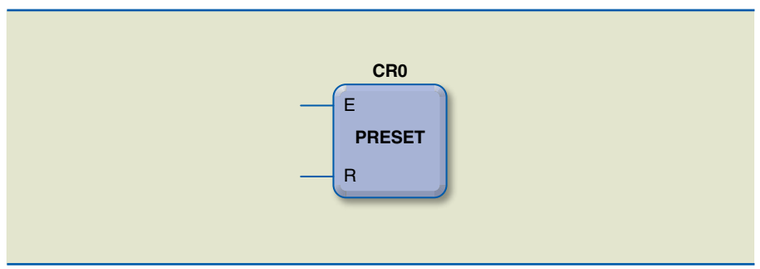
O contador rápido executa a lógica que está programada no CLP em função da entrada rápida. No diagrama Ladder, a entrada rápida tem até oito contadores rápidos, que são habilitados por diferentes bits de entrada e podem executar a contagem de quantidades distintas de pulsos e ser ressetados por diferentes bits.

Nesse contador, a contagem dos pulsos é recebida pela entrada rápida a uma frequência máxima de 4 kHz. A entrada é pelos pinos CR0+ e CR0–. Os pulsos recebidos são incrementados no valor atual da contagem (CRx.V) no bloco do contador rápido.

O funcionamento do contador rápido (figura 5.45) se dá da seguinte maneira: quando a entrada *enable* (E) está habilitada, é feita uma comparação do valor atual (CRx.V) com o valor do *preset* (CRx.P) do bloco do contador rápido. Quando o valor atual for igual ou maior que o valor do *preset*, o bit relacionado ao bloco (CRx.Q) será acionado. Se a entrada *enable* (E) não estiver habilitada, não será efetuada a comparação, mas o valor atual da visualização continuará sendo incrementado a cada pulso recebido na entrada rápida.

O valor atual (CRx.V) do bloco do contador rápido pode ser zerado a qualquer instante com o acionamento do *reset* (R).

Figura 5.45
Símbolo gráfico do contador rápido.



Parâmetros do contador rápido:

- **CRx —** Número do bloco do contador rápido de 0 a 7, definido pelo usuário. Pode haver até oito blocos em apenas um programa.
- **CRx.P —** Valor do *preset* do contador, definido pelo usuário. É um número inteiro na faixa de 0 a 65 535.
- **CRx.V —** Valor atual da contagem, definido por *software*, incrementado a cada pulso recebido pela entrada (E) do contador. É um número inteiro na faixa de 0 a 65 535.
- **R —** *Reset* do contador, definido pelo usuário. Bit (0 ou 1).
- **E:** entrada *enable* do contador, definido pelo usuário. Bit (0 ou 1).
- **CRx.Q —** *Status* de saída do contador rápido. Bit (0 ou 1). Indica que o valor atual do contador rápido (CRx.V) já chegou ao valor do *preset* (CRx.P), acionando essa saída.

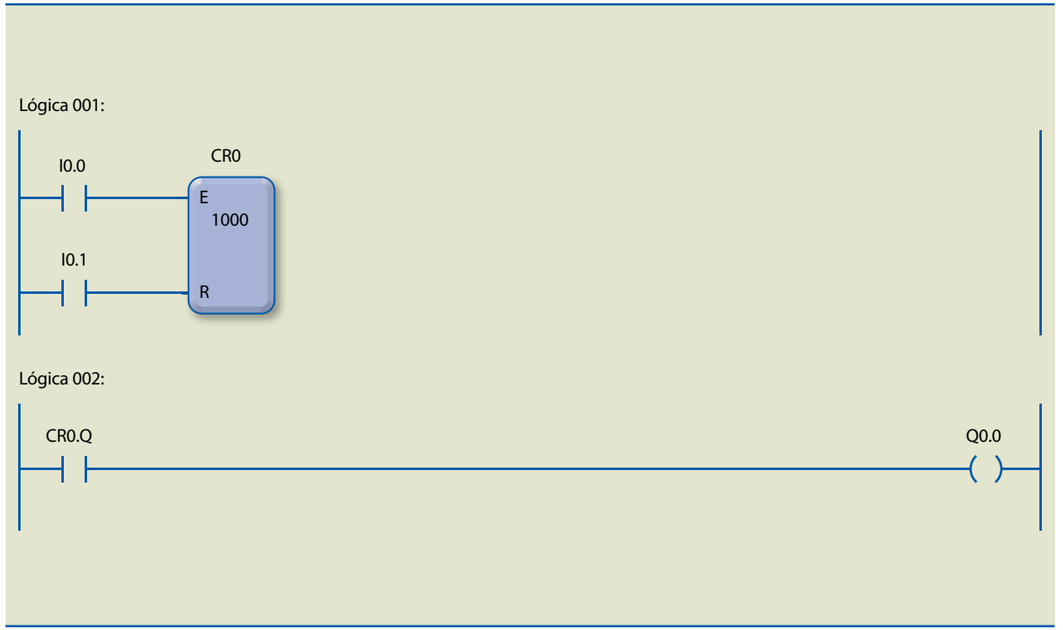
Exemplo

O programa deve fazer a contagem de pulsos da entrada rápida e acionar a saída (Q0.0) quando o contador rápido contar 1 000 pulsos.

Solução:

A figura 5.46 apresenta o programa em diagrama Ladder.

Figura 5.46
Diagrama Ladder.



O programador, ao utilizar os contadores UP, DOWN e Rápido, deve obedecer às regras descritas a seguir.

Na saída dos contadores, os parâmetros aceitos são valores com tamanho de 1 bit (nível lógico “0” ou “1”), ou seja, variáveis também com tamanho de 1 bit. Elas podem ser:

- Qxx (saídas digitais).
- Mxx (bits de memória).
- MRxx (bits de memória retentiva).
- LTxx (LED da IHM).

No valor do *preset* dos contadores, os parâmetros aceitos são valores com tamanho de 1 *word* (valores inteiros de 0 a 65 535).

5.10 Temporizadores

Em sistemas automatizados, é comum incluir a variável tempo no processo. Nesses casos, o temporizador é utilizado para definir o intervalo de tempo entre duas operações, verificar se uma operação ocorre dentro do tempo esperado ou definir o tempo de duração de uma operação. Em geral, podem existir três tipos de temporizadores e em nosso CLP genérico há: temporizador na energização (TON), temporizador na desenergização (TOFF) e temporizador de pulso (TP). Os parâmetros e operandos aqui apresentados são aceitos nesses três tipos.

Ao utilizar os temporizadores TON, TOFF e de pulso, o programador deve obedecer às seguintes regras:

- **Txx —** Número do temporizador, de 0 a 31, definido pelo usuário.
- ***Time base* —** Base de tempo do temporizador (1 s, 0,1 s e 0,01 s), definida pelo usuário.
- **Txx.P —** Valor do *preset* do temporizador, definido pelo usuário. É um número inteiro na faixa de 0 a 65 535.
- **Txx.V —** Valor atual do temporizador, definido por *software*. É um número inteiro na faixa de 0 a 65 535.
- **Q —** *Status* da saída do temporizador, definido por *software*. É ativado quando o valor atual do temporizador se iguala ao valor do preset. Bit (0 ou 1).
- **E —** *Enable* do temporizador, definido pelo usuário. Quando ativado, faz a contagem do tempo. Bit (0 ou 1).

Na saída (Q) do temporizador, os operandos aceitos são aqueles com tamanho de 1 bit (nível lógico 0 ou 1). Essas variáveis podem ser:

- Qxx (saídas digitais).
- Mxx (bits de memória).
- MRxx (bits de memória retentiva).
- LTxx (LED da IHM).

No valor do *preset* do temporizador, os parâmetros aceitos são valores com tamanho de 1 *word* (valor de 0 a 65 535).

5.10.1 Temporizador na energização (TON)

Esse tipo de temporizador (figura 5.47) causa retardo na energização de sua saída. Para isso, ele inicia a contagem do tempo a partir do instante em que a entrada *enable* (E) é habilitada, passando do nível lógico “0” para o “1”. Quando o valor atual do temporizador (Txx.V) se igualar ao tempo do *preset* (Txx.P), a saída do temporizador será acionada, passando do nível lógico “0” para o “1”. Se, a qualquer instante, a entrada *enable* (E) for desabilitada, passando do nível lógico “1” para o “0”, o valor atual do temporizador (Txx.V) será zerado e sua saída (Q) será desabilitada, retornando ao estado inicial, ou seja, nível lógico “0”.

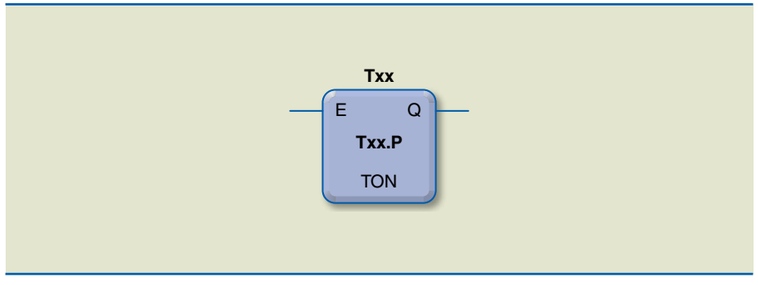


Figura 5.47
Símbolo gráfico do temporizador TON.

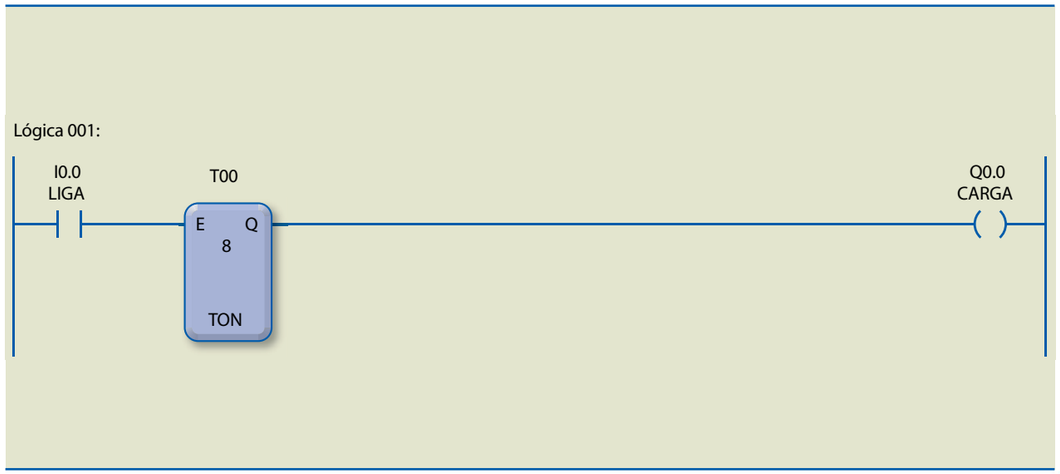
Exemplo

Deseja-se uma programação que acione uma carga que entre em funcionamento 8 segundos depois de o botão liga ser pressionado e desligue em qualquer instante em que o botão for desacionado.

Solução:

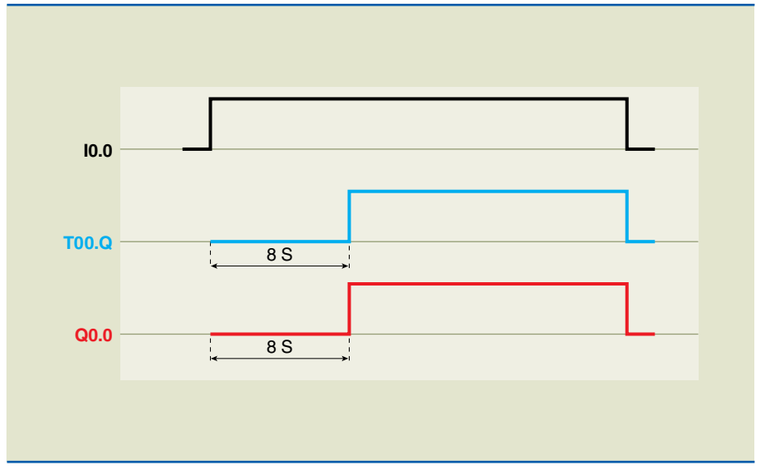
A figura 5.48 apresenta o programa em diagrama Ladder.

Figura 5.48
Diagrama Ladder.



A figura 5.49 mostra o diagrama de estado.

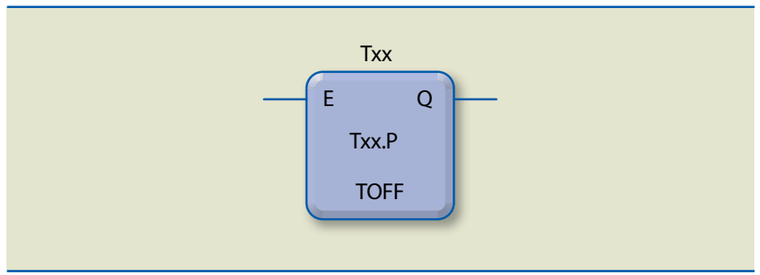
Figura 5.49
Diagrama de estado.



5.10.2 Temporizador na desenergização (TOFF)

Nesse tipo de temporizador (figura 5.50), ao acionarmos a entrada *enable* (E), o valor atual do temporizador (Txx.V) zera e a saída (Txx.Q) passa para o nível lógico "1", acompanhando a entrada. O retardo acontece na desenergização, ou seja, quando desacionamos a entrada *enable* (E), passando do nível lógico "1" para o "0", inicia-se a contagem do tempo que causará o retardo na saída (Txx.Q). A partir do instante em que o valor atual do temporizador (Txx.V) se igualar ao tempo do *preset* (Txx.P), a saída do temporizador será desacionada, passando do nível lógico "1" para o "0".

Figura 5.50
Símbolo gráfico do temporizador TOFF.



Exemplo

Deseja-se desacionar uma carga que interrompa seu funcionamento 8 segundos depois de o botão desliga ser acionado.

Solução:

A figura 5.51 apresenta o programa em diagrama Ladder.

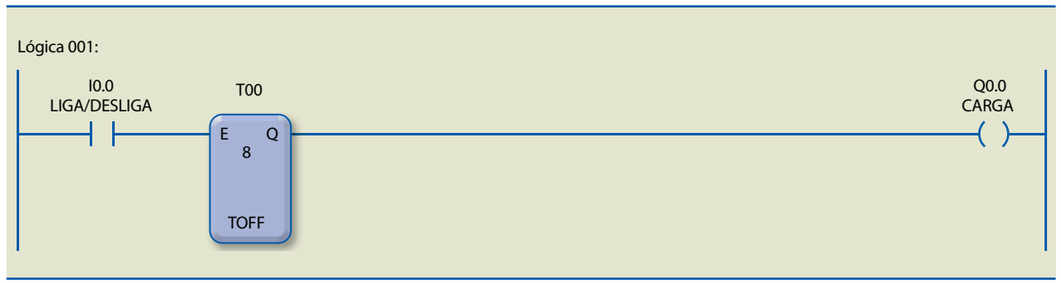


Figura 5.51
Diagrama Ladder.

A figura 5.52 mostra o diagrama de estado.

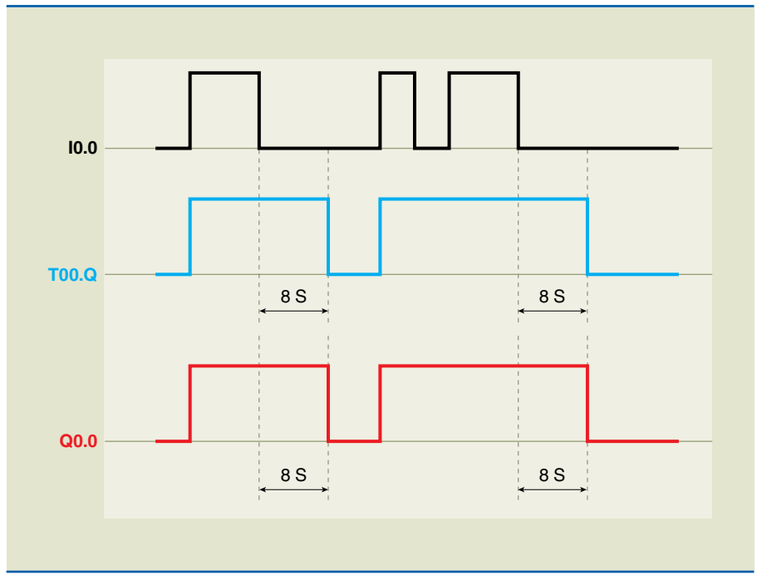


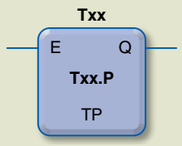
Figura 5.52
Diagrama de estado.

5.10.3 Temporizador de pulso (TP)

Nesse tipo de temporizador (figura 5.53), se, ao acionarmos a entrada *enable* (E), o pulso de entrada for menor que o tempo do *preset* (Txx.P) do temporizador, a saída será igual à entrada. Se a entrada permanecer acionada por tempo maior que o tempo do *preset* (Txx.P) do temporizador, a saída ficará acionada somente pelo tempo do *preset* (Txx.P), gerando um pulso na saída.

A partir do instante em que o valor atual do temporizador (Txx.V) se igualar ao tempo do *preset* (Txx.P), a saída (Q) do temporizador será desativada, passando do nível lógico "1" para o "0". Um fato relevante é que existem pequenas variações no comportamento desse *timer*, dependendo do fabricante. Alguns modelos, por exemplo, mantêm a saída do *timer* ativada mesmo para pulsos curtos no *enable*, desligando após Txx.P. Dessa maneira, recomenda-se consultar o manual do fabricante.

Figura 5.53
Símbolo gráfico do temporizador TP.



O funcionamento desse tipo de instrução pode ser esclarecido no próximo exemplo.

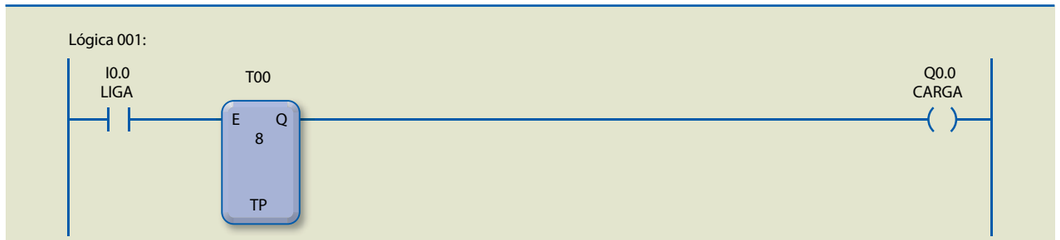
Exemplo

Deseja-se acionar uma carga que interrompa seu funcionamento 8 segundos depois de o botão liga ser acionado. Se desligarmos o botão liga a qualquer instante antes de decorridos os 8 segundos, ele interrompe o funcionamento de imediato.

Solução:

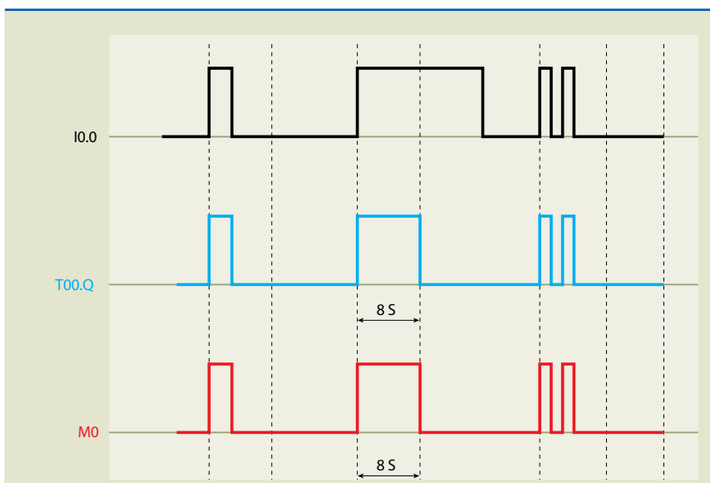
Figura 5.54
Diagrama Ladder.

A figura 5.54 apresenta o programa em diagrama Ladder.



A figura 5.55 mostra o diagrama de estado.

Figura 5.55
Diagrama de estado.



5.11 Entradas e saídas analógicas: endereçamento

Para que trabalhem com sinais analógicos, os CLPs necessitam de um conversor analógico-digital (A/D) nas entradas e, de modo similar, de um conversor digital-analógico (D/A) nas saídas.

A tabela 5.9 apresenta as características das entradas analógicas para o CLP genérico.

Tabela 5.9
Características das entradas analógicas

Entradas analógicas	
IA0	0 a 10 V_{cc} ou 0 a 20 mA
IA1	0 a 10 V_{cc} ou 0 a 20 mA
IA2	0 a 10 V_{cc}
IA3	0 a 10 V_{cc}
Resolução em tensão	12 bits (2,44 mV)
Impedância em tensão	10 kΩ
Resolução em corrente	12 bits (4,8 μA)
Impedância em corrente	500 Ω
Quantidade	4

A tabela 5.10 mostra o endereçamento das entradas analógicas.

Tabela 5.10
Endereçamento das entradas analógicas

Quantidade de entradas	Endereço	Descrição
4	IA0 a IA3	12 bits de resolução referentes às entradas analógicas

A tabela 5.11 apresenta as características das saídas analógicas.

Tabela 5.11
Características das saídas analógicas

Saídas analógicas	
QA0	0 a 10 V_{cc}
QA1	0 a 10 V_{cc}
Resolução	8 bits (39 mV)
Impedância	200 Ω
Quantidade	2

A tabela 5.12 mostra o endereçamento das saídas analógicas.

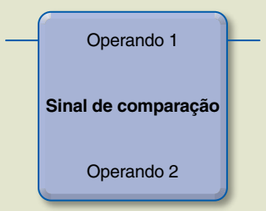
Quantidade de saídas	Endereço	Descrição
2	QA0 e QA1	8 bits de resolução referentes às saídas analógicas

Tabela 5.12
Endereçamento das saídas analógicas

5.12 Instruções de comparação

Em programação, muitas vezes é necessário comparar dois valores. Para isso, pode-se usar as instruções de comparação. Os comparadores utilizam dois operandos, que podem ser byte, *word* ou constante. O programa é realizado com os operandos 1 e 2 (figura 5.56). Caso os valores satisfaçam a condição de comparação e a entrada do comparador esteja habilitada, a saída do comparador será acionada, habilitando, assim, a saída do sistema.

Figura 5.56
Símbolo gráfico do comparador.

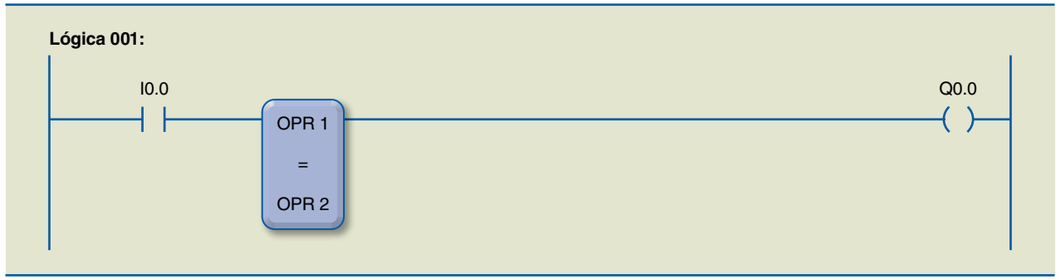


As comparações que podem ser feitas são: igual, maior que, menor que, maior ou igual a, menor ou igual a e diferente.

5.12.1 Igual a (=)

A figura 5.57 apresenta a instrução de comparação igual a (=) em diagrama Ladder.

Figura 5.57
Diagrama Ladder da instrução de comparação igual a (=).

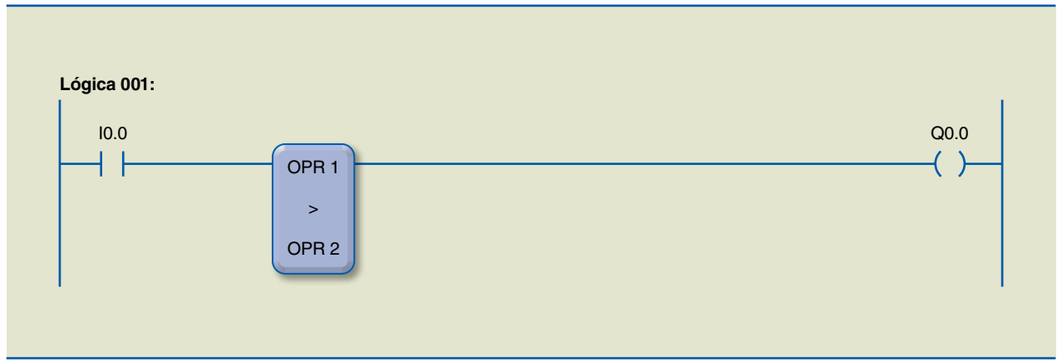


Nesse exemplo, quando a entrada I0.0 estiver habilitada, teremos a comparação entre o operando 1 e o operando 2. Se eles forem iguais, o resultado será nível lógico "1" e a saída será acionada. Se forem diferentes, o resultado será nível lógico "0" e a saída será desligada.

5.12.2 Maior que (>)

A figura 5.58 apresenta o programa da instrução maior que (>) em diagrama Ladder.

Figura 5.58
Diagrama Ladder da instrução maior que (>).

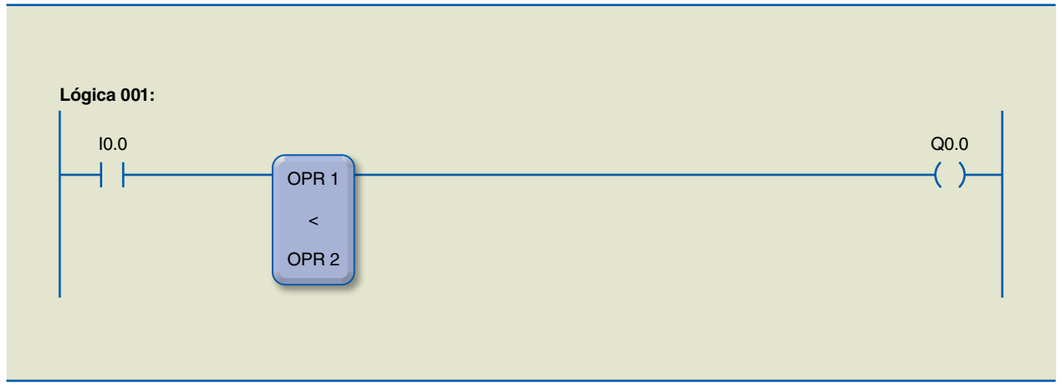


Nesse exemplo, quando a entrada I0.0 estiver habilitada, terá a comparação entre o operando 1 e o operando 2. Se o operando 1 for maior que o operando 2, o resultado terá nível lógico "1" e a saída será acionada. Se o operando 1 for menor que o operando 2, o resultado terá nível lógico "0" e a saída será desligada.

5.12.3 Menor que (<)

A figura 5.59 apresenta o programa da instrução menor que (<) em diagrama Ladder.

Figura 5.59
Diagrama Ladder da instrução menor que (<).

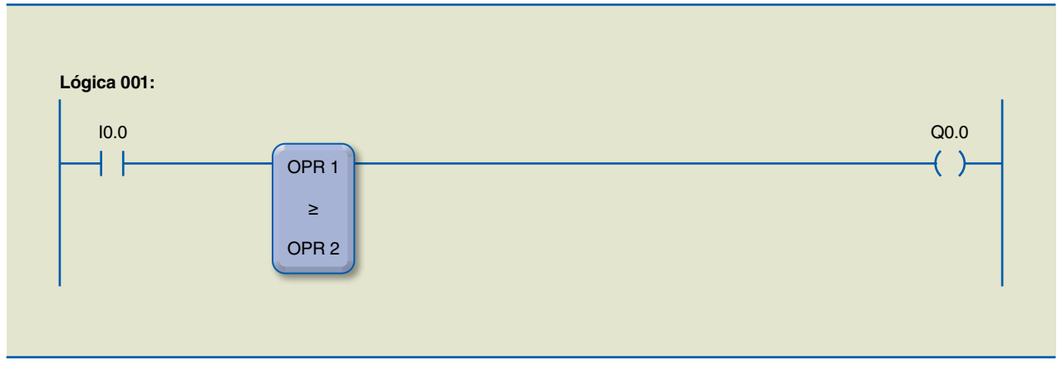


Nesse exemplo, quando a entrada I0.0 estiver habilitada, terá a comparação entre o operando 1 e o operando 2. Se o operando 1 for menor que o operando 2, o resultado terá nível lógico "1" e a saída será acionada. Se o operando 1 for maior ou igual ao operando 2, o resultado terá nível lógico "0" e a saída será desligada.

5.12.4 Maior ou igual a (≥)

Figura 5.60
Diagrama Ladder da instrução maior ou igual a (≥).

A figura 5.60 apresenta o programa da instrução maior ou igual a (≥) em diagrama Ladder.

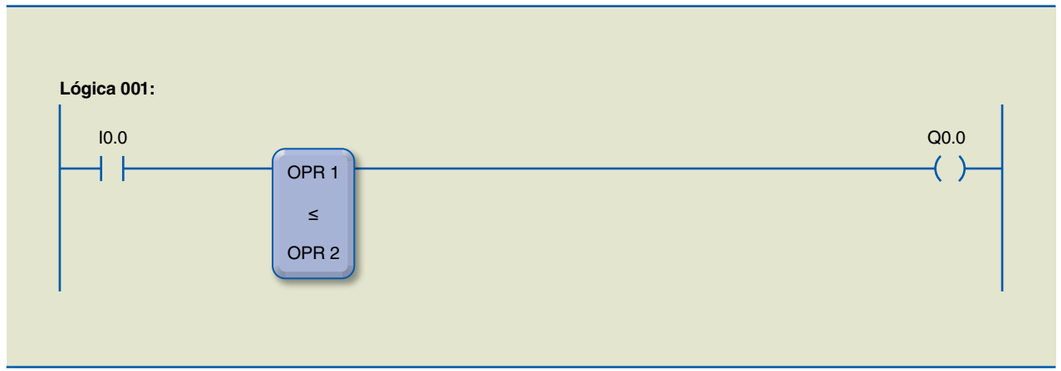


Nesse exemplo, quando a entrada I0.0 estiver habilitada, terá a comparação entre o operando 1 e o operando 2. Se o operando 1 for maior ou igual ao operando 2, o resultado terá nível lógico "1" e a saída será acionada. Se o operando 1 for menor que o operando 2, o resultado terá nível lógico "0" e a saída será desligada.

5.12.5 Menor ou igual a (≤)

Figura 5.61
Diagrama Ladder da instrução menor ou igual a (≤).

A figura 5.61 apresenta o programa da instrução menor ou igual a (≤) em diagrama Ladder.

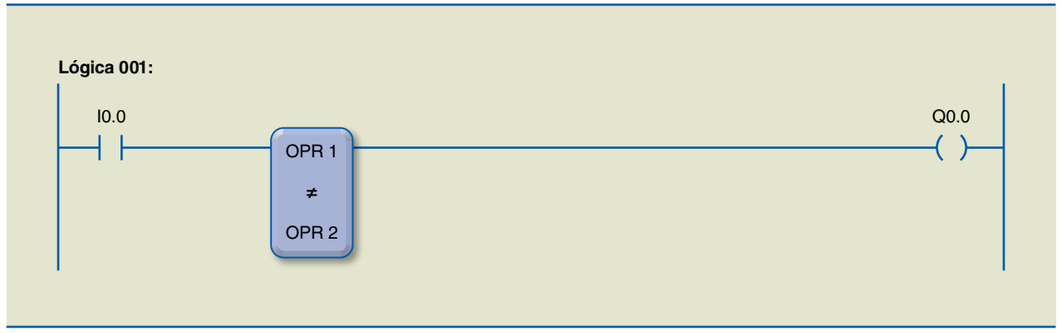


Nesse exemplo, quando a entrada I0.0 estiver habilitada, terá a comparação entre o operando 1 e o operando 2. Se o operando 1 for menor ou igual ao operando 2, o resultado será nível lógico "1" e a saída será acionada. Se o operando 1 for maior que o operando 2, o resultado será nível lógico "0" e a saída será desligada.

5.12.6 Diferente de (≠)

Figura 5.62
Diagrama Ladder da instrução diferente de (≠).

A figura 5.62 apresenta o programa da instrução de comparação diferente de (≠) em diagrama Ladder.



Nesse exemplo, quando a entrada I0.0 estiver habilitada, terá a comparação entre o operando 1 e o operando 2. Se os operandos forem diferentes, o resultado terá nível lógico "1" e a saída será acionada. Se o operando 1 for igual ao operando 2, o resultado terá nível lógico "0" e a saída será desligada.

5.13 Operações matemáticas

Essas instruções têm como função executar operações aritméticas entre dois operandos, colocando o resultado em um operando de resposta denominado RES.

5.13.1 Somador (+)

Quando essa instrução é habilitada por meio da entrada (E), executa-se a soma dos operandos (OPR1 + OPR2), colocando o resultado em RES, conforme demonstra a figura 5.63.

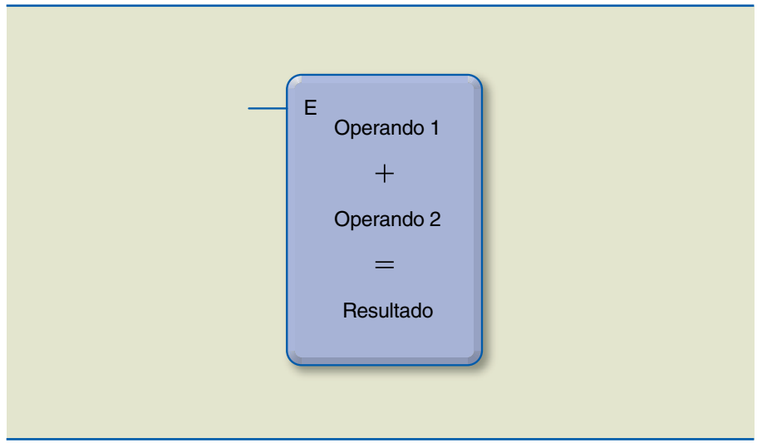


Figura 5.63
Símbolo gráfico do somador.

Quando a entrada está habilitada, tem-se a execução da soma dos dois operandos ou constantes e, consequentemente, o valor passa para um terceiro operando.

Operando 1 e **operando 2** são os valores que serão somados.

Os endereços permitidos são: bytes, *words* e constantes.

Resultado é o operando que receberá o resultado da soma.

5.13.2 Subtrator (–)

Quando essa instrução é habilitada por meio da entrada (E), executa-se a subtração dos operandos (OPR1 – OPR2), colocando o resultado em RES, conforme demonstra a figura 5.64.

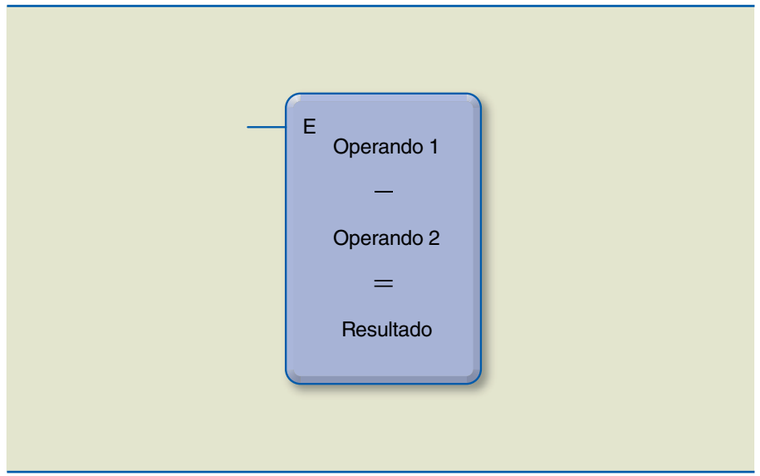


Figura 5.64
Símbolo gráfico do subtrator.

Operando 1 é o valor do qual será subtraído o valor do segundo operando e, consequentemente, o resultado passa para um terceiro operando.

Operando 2 é o valor que será subtraído do primeiro operando.

Os endereços permitidos são: bytes, *words* e constantes.

Resultado é o operando que receberá o resultado da subtração.

Os endereços permitidos são: bytes e *words*.

5.13.3 Multiplicador (×)

Quando essa instrução é habilitada por meio da entrada (E), executa-se a multiplicação dos operandos (OPR1 × OPR2), colocando o resultado em RES, conforme demonstra a figura 5.65.

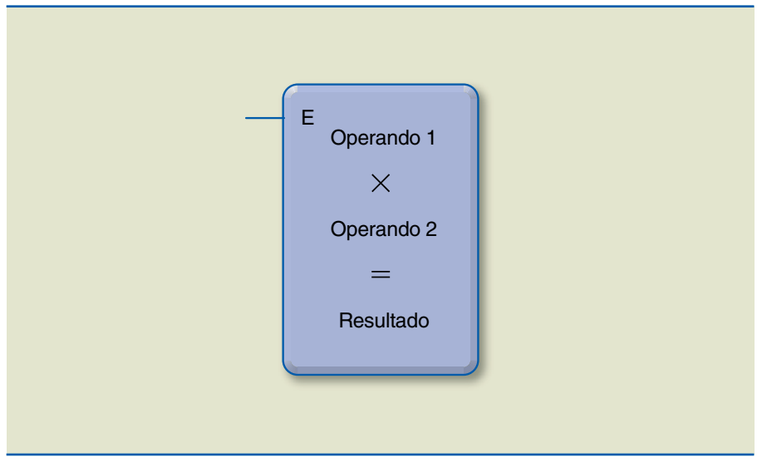


Figura 5.65
Símbolo gráfico do multiplicador.

Quando a entrada está habilitada, tem-se a execução da multiplicação dos dois operandos e, consequentemente, o valor passa para um terceiro operando.

Operando 1 e **operando 2** são os valores que serão multiplicados.

Os endereços permitidos são: bytes, *words* e constantes.

Resultado é o operando que receberá o resultado da multiplicação.

Os endereços permitidos são: bytes e *words*.

5.13.4 Divisor (÷)

Quando essa instrução é habilitada por meio da entrada (E), executa-se a divisão dos operandos (OPR1 ÷ OPR2), colocando o resultado em RES, conforme demonstra a figura 5.66.

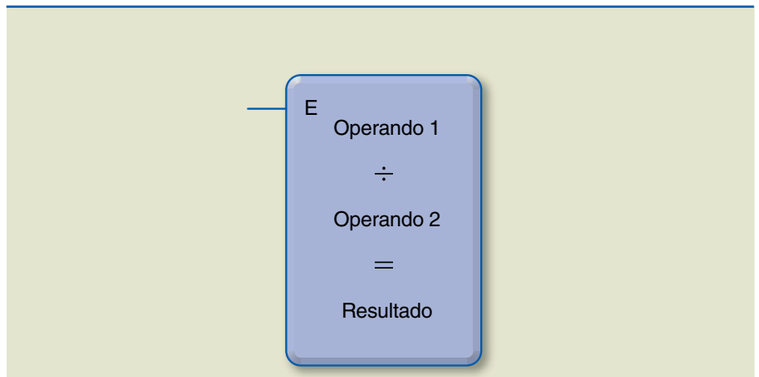


Figura 5.66
Símbolo gráfico do divisor.

Quando a entrada está habilitada, tem-se a execução da divisão dos dois operandos e, consequentemente, o valor passa para um terceiro operando.

Operando 1 e **operando 2** são os valores que serão divididos.

Os endereços permitidos são: bytes, *words* e constantes.

Resultado é o operando que receberá o resultado da divisão.

Os endereços permitidos são: bytes e *words*.

5.14 Funções especiais

5.14.1 CALL

A função especial CALL é mostrada na figura 5.67.

Figura 5.67
Símbolo gráfico da função especial CALL.

Quando essa instrução é habilitada, o programa executa a sub-rotina indicada em CALL e, após a execução, retorna para o mesmo ponto do programa que chamou a sub-rotina.

5.14.2 JUMP

A função especial JUMP é mostrada na figura 5.68.

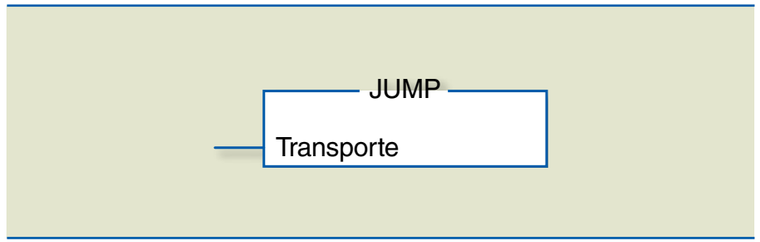


Figura 5.68
Símbolo gráfico da função especial JUMP.

Quando essa instrução é habilitada, o programa, ao passar pela instrução, pula para a sub-rotina indicada em JUMP. Nessa instrução, não há retorno para a rotina que estava sendo executada; o programa continua na sub-rotina indicada.

5.14.3 MOVE

A figura 5.69 mostra a função especial MOVE.

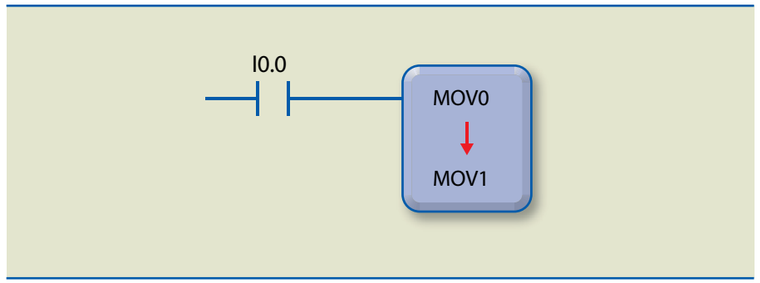


Figura 5.69
Símbolo gráfico da função especial MOVE.

Essa instrução possui duas variáveis: MOV0 (origem) e MOV1 (destino). Quando a entrada I0.0 passa do nível lógico "0" para o "1", a instrução MOVE é habilitada, transferindo o valor contido na variável MOV0 (origem) para a variável MOV1 (destino).