Programação de CLPs Métodos e Técnicas

Volume 1

Rafael Manfrin Mendes

EDITORA SCIENZA

Rafael Manfrin Mendes

Programação de CLPs
Métodos e Técnicas

EDITORA SCIENZA
2021

M5226p Mendes, Rafael Manfrin

Programação de CLPs. Métodos e Técnicas / Rafael Manfrin Mendes. São Carlos, 2021.

286 p.

ISBN 978-65-5668-037-8

1. Controladores Lógico Programáveis. 2. Estratégias de programação. 3. Linguagens de programação. 4. Técnicas de implementação. I. Autor. II. Título.

CDD 600

Revisão, Editoração, E-book e Impressão:

EDITORA SCIENZA

Rua Juca Sabino, 21 – São Carlos, SP

(16) 9 9285-3689

www.editorascienza.com.br

gustavo@editorascienza.com

Sumário

Capítulo 1

Controlador Lógico Programável

1.1 – Estrutura básica do CLP

1.1.1 – Introdução

A engenharia de controle evoluiu com o tempo. No passado, os humanos eram o principal método para controlar um sistema. Mais recentemente, a eletricidade foi usada para controle e o controle elétrico inicial baseava-se em relés. Esses relés permitem que a alimentação seja ligada e desligada sem um interruptor mecânico. É comum usar relés para tomar decisões de controle lógico simples. O desenvolvimento do computador de baixo custo trouxe a mais recente revolução, o Controlador Lógico Programável (CLP). O advento do CLP começou na década de 1970 e se tornou a escolha mais comum para controles de fabricação.

Os CLPs vêm ganhando popularidade no chão de fábrica e provavelmente permanecerão predominantes por algum tempo. A maior parte disso se deve às vantagens que eles oferecem.

- Eficaz na redução de custos para controlar sistemas complexos.
- Flexível e pode ser reaplicado para controlar outros sistemas de forma rápida e fácil.
- Habilidades computacionais permitem um controle mais sofisticado.
- Os recursos de solução de problemas tornam a programação mais fácil e reduzem o tempo de inatividade.
- Componentes confiáveis tornam provável que operem por anos antes de falhar.

1.1.2 – Lógica Ladder

A lógica Ladder é o principal método de programação usado no CLP. A lógica ladder foi desenvolvida para simular a lógica do relé. A decisão de usar os diagramas lógicos do relé foi estratégica. Ao selecionar a lógica ladder como o principal método de programação, a quantidade de retreinamento necessária para engenheiros e comerciantes foi bastante reduzida.

Os sistemas de controle modernos ainda incluem relés, mas raramente são usados para lógica. Um relé é um dispositivo simples que usa um campo magnético para controlar uma chave, conforme ilustrado na Figura 1.1. Quando uma tensão é aplicada à bobina de entrada, a corrente resultante cria um campo magnético. O campo magnético puxa uma chave de metal (ou palheta) em sua direção e os contatos se tocam, fechando a chave. O contato que fecha quando a bobina é energizada é denominado normalmente aberto (N.A.). Os contatos normalmente fechados (N.F.) se tocam quando a bobina de entrada não está energizada. Os relés são normalmente desenhados de forma esquemática usando um círculo para representar a bobina de entrada. Os contatos de saída são mostrados com duas linhas paralelas. Os contatos normalmente abertos são mostrados como duas linhas e estarão abertos (não conduzindo) quando a entrada não estiver energizada. Os contatos normalmente fechados são mostrados com duas linhas com uma linha diagonal através deles. Quando a bobina de entrada não é energizada, os contatos normalmente fechados estão fechados (conduzindo).

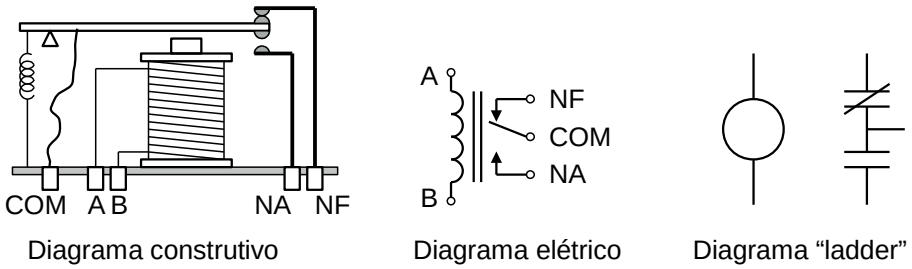

Figura 1.1 – Diagramas esquemáticos de um relé.

Os relés são usados para permitir que uma fonte de alimentação feche uma chave para outra fonte de alimentação (geralmente de alta corrente), enquanto os mantém isolados. Um exemplo de um relé em uma aplicação de controle simples é mostrado na Figura 1.2. Neste sistema, o primeiro relé à esquerda é usado normalmente fechado e permitirá que a corrente flua até que uma tensão seja aplicada à entrada A. O segundo relé está normalmente aberto e não permitirá que a corrente flua até que uma tensão seja aplicada à entrada B. Se a corrente estiver fluindo pelos dois primeiros relés, então a corrente fluirá pela bobina no terceiro relé e fechará a chave para a saída C. Este circuito normalmente seria desenhado na forma de lógica ladder. Isso pode ser lido logicamente, pois C estará ativado se A estiver desativado e B estiver ativado.

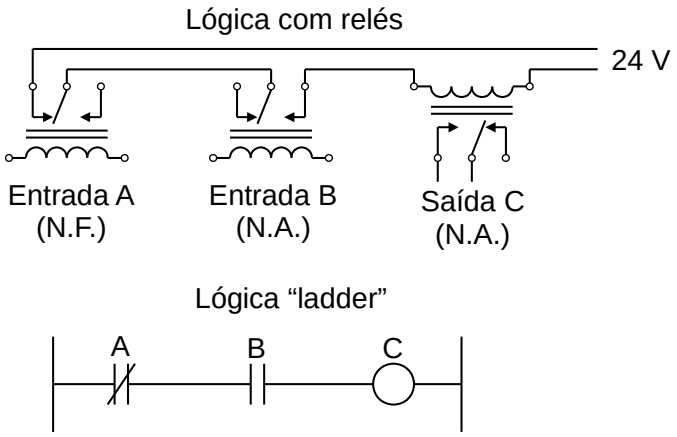

Figura 1.2 – Um controle simples com relé.

O exemplo da Figura 1.2 não mostra todo o sistema de controle, mas apenas a lógica. Quando consideramos um CLP, existem entradas, saídas e a lógica. A Figura 1.3 mostra uma representação mais completa do CLP. Aqui, existem duas entradas de botões. Podemos imaginar as entradas ativando bobinas de relé de 24 Vcc no CLP. Isso, por sua vez, aciona um relé de saída que liga 115 Vca, que acenderá uma luz. Observe que, em CLPs reais, as entradas nunca são relés, mas as saídas geralmente são relés. A lógica ladder no CLP é, na verdade, um programa de computador que o usuário pode inserir e alterar. Observe que ambos os botões de entrada estão normalmente abertos, mas a lógica ladder dentro do CLP tem um contato normalmente aberto e um contato normalmente fechado. Não pense que a lógica ladder no CLP precisa corresponder às entradas ou saídas. Muitos iniciantes serão pegos tentando fazer a lógica ladder corresponder aos tipos de entrada.

Muitos relés também têm várias saídas (contatos) e isso permite que um relé de saída também seja uma entrada simultaneamente. O circuito mostrado na Figura 1.4 é um exemplo disso, é chamado de selo no circuito. Neste circuito, a corrente pode fluir através de qualquer um dos ramos do circuito, através dos contatos marcados com A ou B. A entrada B só estará ligada quando a saída B estiver ligada. Se B estiver desligado e A estiver energizado, então B será ligado. Se B for ativado, a entrada B será

ativada e manterá a saída B ativada mesmo que a entrada A seja desativada. Depois que B for ligado, a saída B não será desligada.

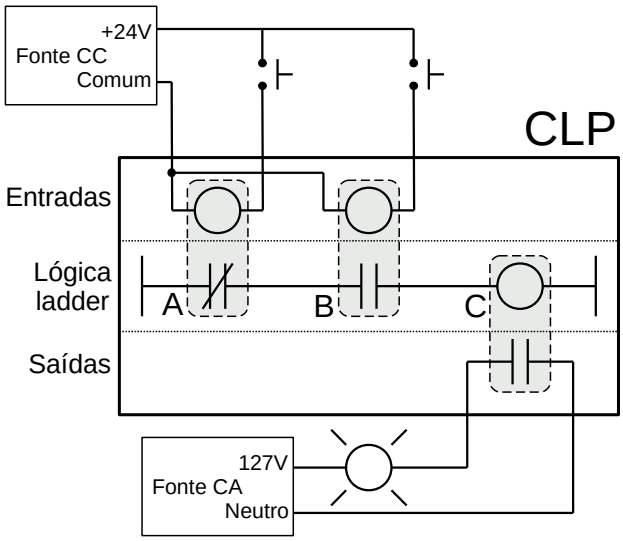

Figura 1.3 – Um CLP com relés.

Figura 1.4 – Um circuito de selo.

Observação: quando A é pressionado, a saída B liga e a entrada B também liga e mantém B ligado permanentemente, até que a alimentação seja removida.

Nota: A linha à direita foi removida intencionalmente e está implícita nesses diagramas.

1.1.3 – Programação

Os primeiros CLPs foram programados com uma técnica baseada em esquemas de cabeamento lógico de relé. Isso eliminou a necessidade de ensinar eletricistas, técnicos e engenheiros a programar um computador – mas esse método "pegou" e é a técnica mais comum para programação de CLPs atualmente. Um exemplo de lógica ladder pode ser visto na Figura 1.5. Para interpretar este diagrama, imagine que a alimentação de energia está na linha vertical do lado esquerdo, chamamos isso de barramento quente. No lado direito está o trilho neutro. Na figura, existem dois degraus, e em cada um deles há combinações de entradas (duas linhas verticais) e saídas (círculos). Se as entradas forem abertas ou fechadas na combinação certa, a energia pode fluir do barramento quente, através das entradas, para alimentar as saídas e, finalmente, para o barramento neutro. Uma entrada pode vir de um sensor, chave ou qualquer outro tipo de sensor. Uma saída será algum dispositivo fora do CLP que será ligado ou desligado, como luzes ou motores. No degrau superior (001), os contatos estão normalmente abertos e normalmente fechados. O que significa que se a entrada A estiver ligada e a entrada B desligada, a energia fluirá pela saída e a ativará. Qualquer outra combinação de valores de entrada resultará na saída X desligada.

A segunda linha (002) da Figura 1.5 é mais complexa; na verdade, existem várias combinações de entradas que resultarão na saída Y ativada. Na parte mais à esquerda da linha, a energia pode fluir pela parte superior se C estiver desligado e D estiver ligado. A energia também pode (e simultaneamente) fluir pela parte inferior se E e F forem verdadeiros. Isso obteria energia no meio do degrau e, então, se G ou H for verdadeiro, a energia será fornecida à saída Y.

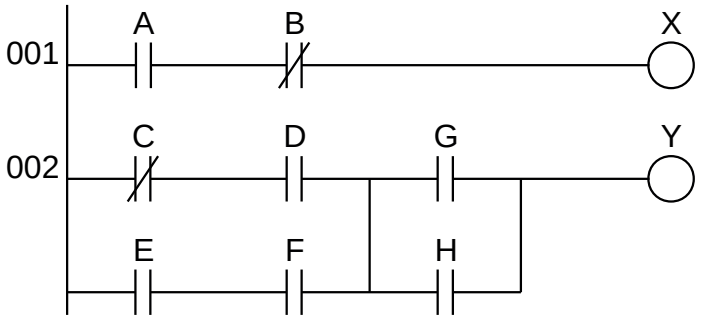

Figura 1.5 – Um diagrama simples de lógica ladder.

Nota: A energia precisa fluir através de alguma combinação das entradas (A, B, C, D, E, F, G, H) para ligar as saídas (X, Y).

Existem outros métodos de programação de CLPs. Uma das primeiras técnicas envolvia instruções mnemônicas. Essas instruções podem ser derivadas diretamente dos diagramas de lógica ladder e inseridas no CLP por meio de um terminal de programação simples. Um exemplo de mnemônicos é mostrado na Figura 1.6. Neste exemplo, as instruções são lidas uma linha por vez, de cima para baixo. A primeira linha 0000 tem a instrução LDN (carga de entrada e não) para a entrada A. Isso examinará a entrada para o CLP e se estiver desligada lembrará de 1 (ou verdadeiro), se estiver ligada lembrará de 0 (ou falso). A próxima linha usa uma instrução LD (carga de entrada) para examinar a entrada. Se a entrada estiver desligada, ele se lembra de um 0, se a entrada estiver ligada, ele se lembra de um 1 (nota: este é o reverso do LDN). A instrução AND recupera os dois últimos números lembrados e, se ambos forem verdadeiros, o resultado é 1, caso contrário, o resultado é 0. Este resultado agora substitui os dois números que foram lembrados e há apenas um número lembrado. O processo é repetido para as linhas 0003 e 0004, mas quando isso é feito, agora três números são lembrados. O número mais antigo é do AND, os números mais novos são das duas instruções LD. O AND na linha 0005 combina os resultados das últimas instruções LD e agora há dois números lembrados. A instrução OR leva os dois números restantes e se um deles for 1, o resultado será 1, caso contrário, o resultado será 0. Este resultado substitui os dois números e agora há um único número ali. A última instrução é o ST (armazenar saída) que vai olhar o último valor armazenado e se for 1, a saída será ligada, se for 0 a saída será desligada.

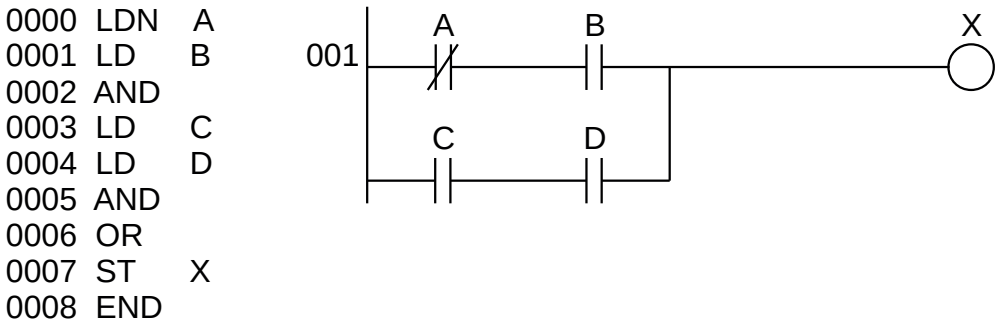

Figura 1.6 – Um exemplo de programa mnemônico e lógica ladder equivalente.

O programa de lógica ladder na Figura 1.6 é equivalente ao programa mnemônico. Mesmo que você tenha programado um CLP com lógica ladder, ele será convertido para a forma mnemônica antes de ser usado pelo CLP. No passado, a programação mnemônica era a mais comum, mas agora é incomum que os usuários vejam programas mnemônicos.

Os gráficos de funções sequenciais (SFCs) foram desenvolvidos para acomodar a programação de sistemas mais avançados. Eles são semelhantes aos fluxogramas, mas muito mais poderosos. O exemplo visto na Figura 1.7 está fazendo duas coisas diferentes. Para ler o gráfico, comece no topo, onde diz "início". Abaixo disso, há uma linha horizontal dupla que diz seguir os dois caminhos. Como resultado, o CLP começará a seguir o ramo nos lados esquerdo e direito separadamente e simultaneamente. À esquerda, há duas funções, a primeira é a função de "ligar". Esta função será executada até decidir que está feito, e a função de "desligar" virá depois. No lado direito está a função "rápido", que funcionará até que seja concluída. Essas funções parecem inexplicáveis, mas cada função, como ligar, será um pequeno programa de lógica ladder. Esse método é muito diferente dos fluxogramas porque não precisa seguir um único caminho por meio do fluxograma.

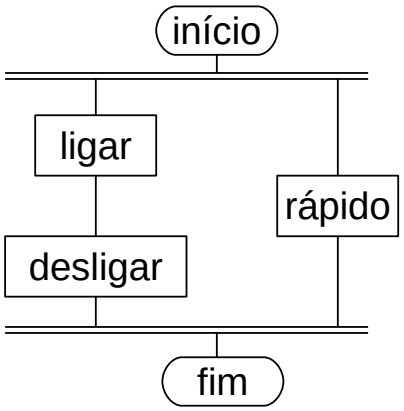

Figura 1.7 – Um exemplo de um gráfico de função sequencial.

A programação por Texto Estruturado foi desenvolvida como uma linguagem de programação mais moderna. É bastante semelhante a linguagens como BASIC. Um exemplo simples é mostrado na Figura 1.8. Este exemplo usa um local de memória do PLC com nome "i". Esta localização da memória é para um número inteiro. A primeira linha do programa define o valor como 0. A próxima linha começa um laço ("loop") e será para onde o laço retorna. A próxima linha recupera o valor no local "i", adiciona 1 a ele e o retorna ao mesmo local. A próxima linha verifica se o laço deve ser encerrado. Se "i" for maior ou igual a 10, o laço será encerrado, caso contrário, o computador retornará para a instrução REPEAT e continuará a partir daí. Cada vez que o programa passa por este ciclo, "i" aumentará em 1 até que o valor alcance 10.

```
i:=0;
REPEAT
i:=i+1;
UNTIL i>=10
END_REPEAT
```

Figura 1.8 – Um exemplo de um programa de texto estruturado.

1.1.4 – Conexões no CLP

Quando um processo é controlado por um CLP, ele usa as entradas dos sensores para tomar decisões e atualizar as saídas para os atuadores de acionamento, conforme mostrado na Figura 1.9. O processo é um processo real que mudará com o tempo. Os atuadores conduzirão o sistema a novos estados (ou modos de operação). Isso significa que o controlador é limitado pelos sensores disponíveis, se uma entrada não estiver disponível, o controlador não terá como detectar uma condição.

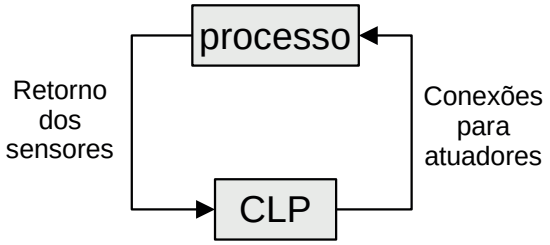

Figura 1.9 – A separação de controlador e processo.

A malha de controle é um ciclo contínuo das entradas de leitura do CLP, resolvendo a lógica ladder e, em seguida, alterando as saídas. Como qualquer computador, isso não acontece instantaneamente. A Figura 1.10 mostra o ciclo básico de operação de um CLP ("scan"). Quando a alimentação é ligada inicialmente, o CLP faz uma verificação rápida de integridade para garantir que o hardware está funcionando corretamente. Se houver um problema, o CLP irá parar e indicar que há um erro. Por exemplo, se a energia do CLP estiver caindo e prestes a desligar, isso resultará em um tipo de falha. Se o CLP passar na verificação de integridade, ele fará a varredura (leitura) todas as entradas. Depois que os valores das entradas são armazenados na memória, a lógica ladder será varrida (resolvida) usando os valores armazenados - não os valores atuais. Isso é feito para evitar problemas de lógica quando as entradas mudam durante a varredura de lógica ladder. Quando a varredura da lógica ladder for concluída, as saídas serão varridas (os valores de saída serão alterados). Depois disso, o sistema volta para fazer uma verificação de integridade e o loop continua indefinidamente. Ao contrário dos computadores normais, todo o programa será executado a cada verificação. Os tempos típicos para cada um dos estágios são da ordem de milissegundos.

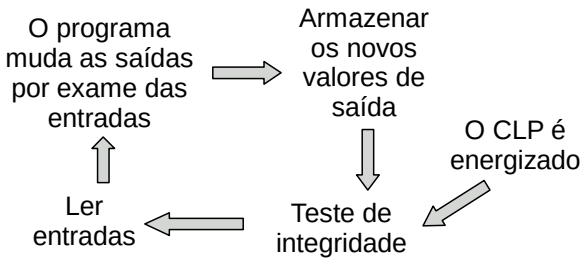

Figura 1.10 – O ciclo básico de operação do CLP.

1.2 – Estrutura física do CLP

1.2.1 – Introdução

Muitas configurações de CLP estão disponíveis, até mesmo de um único fornecedor. Mas, em cada um deles existem componentes e conceitos comuns. Na Figura 1.11 pode ser visto o diagrama básico dos diversos componentes do CLP. Os componentes mais essenciais são:

- Fonte de alimentação – Pode ser incorporada ao CLP ou ser uma unidade externa. Os níveis de tensão comuns exigidos pelo CLP (com e sem a fonte de alimentação) são 24Vdc, 120Vac, 220Vac.
- CPU (Central Processing Unit) – Este é um computador onde a lógica ladder é armazenada e processada.
- Entrada e Saída – Vários terminais de entrada e saída devem ser fornecidos para que o CLP possa monitorar o processo e iniciar ações.
- Luzes indicadoras – Indicam o status do CLP, incluindo alimentação, programa em execução e falha. Eles são essenciais ao diagnosticar problemas.
- Base ou Rack – Conexão mecânica e elétrica. Barramento de comunicação e alimentação.

A configuração do CLP refere-se à embalagem dos componentes. As configurações típicas estão listadas abaixo, da maior para a menor.

- Rack – um rack geralmente é grande e pode conter vários cartões. Quando necessário, vários racks podem ser conectados juntos. Estes tendem a ser os de maior custo, mas também os mais flexíveis e fáceis de manter.
- Mini – são menores do que os racks de CLP de tamanho normal, mas podem ter a mesma capacidade de E/S.
- Micro – essas unidades podem ser tão pequenas quanto um baralho de cartas. Eles tendem a ter quantidades fixas de E/S e habilidades limitadas, mas os custos serão os mais baixos.
- Software – Um CLP baseado em software requer um computador com uma placa de interface, mas permite que o CLP seja conectado a sensores e outros CLPs em uma rede.

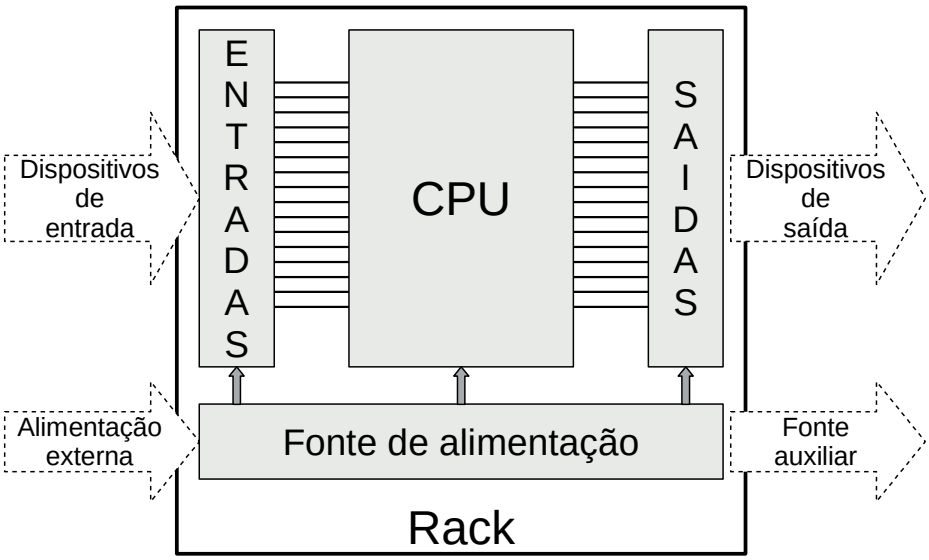

Figura 1.11 – Estrutura física do CLP.

1.2.2 – CPU

A CPU é composta de circuitos eletrônicos para a realização das tarefas de execução do programa e interligação com os dispositivos externos. É composta de microprocessador ou microcontrolador, memórias, dispositivos de relógio em tempo real, temporizadores autônomos e barramentos digitais para acesso a dispositivos periféricos.

Processador

Microprocessador/controlador convencional - 80286, 80386, 8051, ou processador dedicado DSP (Processador Digital de Sinais). Há CPUs com processamento paralelo (sistema de redundância) para execução do mesmo programa de aplicação e confronto de resultados. Algumas famílias com Coprocessadores para auxílio em operações complexas. A operação do processador são: controle de barramentos; interpreta e executa as instruções do programa de aplicação; controla comunicação com dispositivos externos; verifica integridade de todo sistema (diagnósticos). Pode operar com registros e palavras de instrução ou dados – 8,16 e 32 bits. Atualmente é utilizado microcontroladores integrados.

Memória

Composto por: Memória de Operação e Memória de Aplicação.

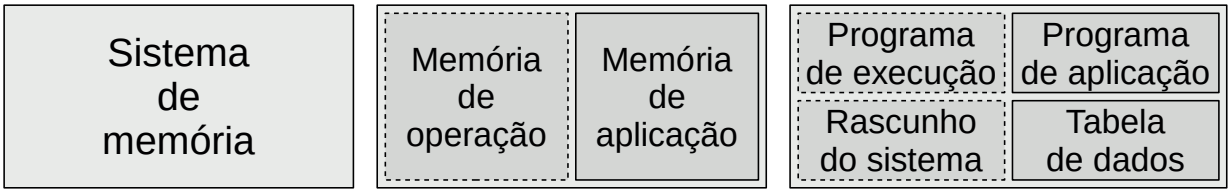

Figura 1.12 – Sistema de memória do CLP.

Memória do Sistema de Operação

Programa de execução (Firmware)

- Programa do fabricante - Controle de operação do sistema; execução do Programa de Aplicação; controle de serviços periféricos, atualização dos módulos de E/S.
- Tradução do Programa de Aplicação de linguagem de alto nível para linguagem de máquina.
- Armazenado em EPROM (normalmente).

Rascunho do sistema

- Armazenamento temporário de pequena quantidade de dados utilizados pelo Sistema de Operação: calendário, relógios internos, marcas de alarmes e erros, marcadores internos, valores de contadores e temporizadores, registradores para operações matemáticas.
- Memória tipo RAM.

Memória de aplicação ou memória do usuário

Programa de Aplicação

- Programa do usuário – memória EEPROM (normalmente) ou ainda EPROM ou RAM com bateria de segurança.

Tabela de dados

- Valores atuais e de preset de temporizadores, contadores e variáveis do programa;
- Status dos pontos de entrada e saída (tabela imagem das entradas e saídas) - cada um conectado aos módulos de E/S com endereços específicos na tabela de dados.
- Memória do tipo RAM - podendo ser alimentada com bateria.

1.2.3 – Entradas e saídas

As entradas e saídas de um CLP são necessárias para monitorar e controlar um processo. Tanto as entradas quanto as saídas podem ser categorizadas em dois tipos básicos: lógico ou contínuo. Considere o exemplo de uma lâmpada. Se só puder ser ligado ou desligado, é um controle lógico. Se a luz pode ser reduzida para níveis diferentes, ela é contínua. Os valores contínuos parecem mais intuitivos, mas os valores lógicos são preferidos porque permitem mais certeza e simplificam o controle. Como resultado, a maioria dos aplicativos de controle (e CLPs) usam entradas e saídas lógicas para a maioria dos aplicativos.

As saídas para os atuadores permitem que um CLP faça com que algo aconteça em um processo. Uma pequena lista de atuadores populares é fornecida abaixo em ordem de popularidade relativa.

- Válvulas solenoides – saídas lógicas que podem alternar um fluxo hidráulico ou pneumático.
- Luzes – saídas lógicas que geralmente podem ser alimentadas diretamente das placas de saída do CLP.
- Partidas de motor – os motores frequentemente consomem uma grande quantidade de corrente quando dão partida, portanto, exigem partidas de motor, que são basicamente relés grandes.
- Servos motores – uma saída contínua do CLP pode comandar uma velocidade ou posição variável.

As saídas dos CLPs geralmente são relés, mas também podem ser eletrônicos de estado sólido, como transistores para saídas CC ou Triacs para saídas CA. As saídas contínuas requerem cartões de saída especiais com conversores digital para analógico.

As entradas vêm de sensores que traduzem fenômenos físicos em sinais elétricos. Exemplos típicos de sensores estão listados abaixo em ordem relativa de popularidade.

- Chaves de proximidade – usa indutância, capacitância ou luz para detectar um objeto logicamente.
- Interruptores – os mecanismos mecânicos abrirão ou fecharão os contatos elétricos para um sinal lógico.
- Potenciômetro – mede posições angulares continuamente, usando resistência.
- LVDT (transformador diferencial linear variável) - mede o deslocamento linear continuamente usando acoplamento magnético.

As entradas para um CLP vêm em algumas variedades básicas, as mais simples são as entradas CA e CC. AS entradas do tipo fornecedoras (“Sourcing”) e consumidoras (“Sinking”) também são populares. Este método de saída determina que um dispositivo não forneça potência. Em vez disso, o dispositivo apenas liga ou desliga a corrente, como um simples interruptor.

- Fornecedora – Quando ativa, a saída permite que a corrente flua para um aterramento comum. É a melhor seleção quando diferentes tensões são fornecidas.
- Consumidora – Quando ativo, a corrente flui de uma fonte, através do dispositivo de saída e para o aterramento. Este método é melhor usado quando todos os dispositivos usam uma única tensão de alimentação.

Isso também é conhecido como NPN (consumidora) e PNP (fornecedora). PNP é mais popular. Isso será abordado em detalhes no capítulo sobre sensores.

Entradas

Em CLPs menores, as entradas são normalmente integradas e são especificadas no momento da compra do CLP. Para CLPs maiores, as entradas são adquiridas como módulos, ou cartões, com 8 ou 16 entradas do mesmo tipo em cada cartão. Para fins de discussão, discutiremos todas as entradas como se tivessem sido compradas como cartões. A lista abaixo mostra faixas típicas para tensões de entrada e está aproximadamente em ordem de popularidade.

12-24 Vcc
100-120 Vca
10-60 Vcc
12-24 Vca/cc
5 Vcc (TTL)
200-240 Vca
48 Vcc
24 Vca

As placas de entrada de CLP raramente fornecem energia, isso significa que uma fonte de alimentação externa é necessária para fornecer energia para as entradas e sensores. O exemplo na Figura 1.13 mostra como conectar uma placa de entrada CA.

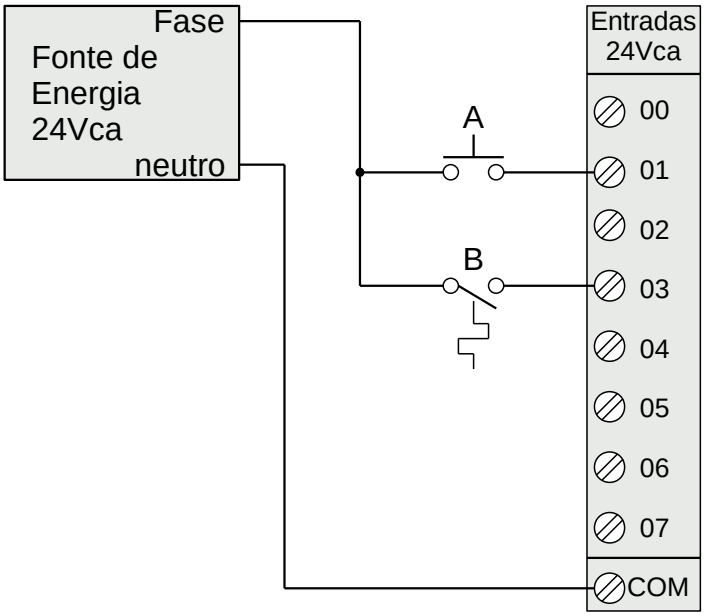

Figura 1.13 – Um cartão de entrada AC.

No exemplo, há duas entradas, uma é um botão normalmente aberto (A) e a segunda é um interruptor de temperatura ou relé térmico (B). Ambos os interruptores são alimentados pela saída positiva/fase da fonte de alimentação de 24 Vca – é como se fosse o terminal positivo em uma fonte de alimentação CC.

A energia é fornecida ao lado esquerdo de ambos os interruptores. Quando os interruptores estão abertos, não há passagem de tensão para a placa de entrada. Se qualquer um dos interruptores estiver fechado, a energia será fornecida à placa de entrada. Neste caso, as entradas 1 e 3 são usadas – observe que as entradas começam em 0. A placa de entrada compara essas tensões com as comuns. Se a tensão de entrada estiver dentro de uma determinada faixa de tolerância, as entradas serão ativadas.

Nota: O processo de projeto será muito mais fácil se as entradas e saídas forem planejadas primeiro e as etiquetas dos dispositivos forem inseridos antes da lógica ladder. Em seguida, o programa é inserido usando os nomes das etiquetas e é muito mais simples.

Muitos iniciantes ficam confusos sobre onde as conexões são necessárias no circuito acima. A palavra-chave a lembrar é *circuito*, o que significa que há um laço completo que a tensão deve ser capaz de seguir. Na Figura 1.11 podemos começar a seguir o circuito (laço) na fonte de alimentação. O caminho passa pelos interruptores, pela placa de entrada e volta para a fonte de alimentação, onde flui de volta para o início. Em uma implementação de CLP completa, haverá muitos circuitos que devem ser concluídos cada um.

Um segundo conceito importante é o *comum*. Aqui, o neutro da fonte de alimentação é a tensão comum ou de referência. Na verdade, escolhemos esta como nossa referência de 0 V e todas as outras tensões são medidas em relação a ela. Se tivéssemos uma segunda fonte de alimentação, também precisaríamos conectar o neutro para que ambos os neutros fossem conectados ao mesmo comum. Frequentemente, o comum e o aterramento serão confundidos. O comum é uma referência que é usada para 0 V, mas o aterramento é usado para evitar choques e danos ao equipamento. O aterramento é conectado sob um edifício a um tubo de metal ou grade no solo. Este é conectado ao sistema elétrico de um edifício, às tomadas, onde são conectadas as caixas metálicas dos equipamentos elétricos. Quando a energia flui pelo solo, é ruim. Infelizmente, muitos engenheiros e fabricantes misturam o aterramento e o comum. É muito comum encontrar uma fonte de alimentação com o aterramento e comum erroneamente etiquetada.

Lembre-se: não misture o aterramento com o comum. Não os conecte se o comum de seu dispositivo estiver conectado a um comum em outro dispositivo.

Um conceito final que tende a prender os iniciantes é que cada placa de entrada é isolada. Isso significa que se você conectou um comum a apenas uma placa, as outras placas não estão conectadas. Quando isso acontece, os outros cartões não funcionam corretamente. Você deve conectar um comum para cada uma das placas de saída. Existem muitas compensações ao decidir que tipo de placa de entrada usar.

- As tensões CC são geralmente mais baixas e, portanto, mais seguras (ou seja, 12-24 V).
- As entradas CC são muito rápidas, as entradas CA requerem um tempo de ativação mais longo. Por exemplo, uma onda de 60 Hz pode exigir até 1/60 seg para um reconhecimento razoável.
- Tensões CC podem ser conectadas a uma grande variedade de sistemas elétricos.
- Os sinais CA são mais imunes a ruídos do que CC, portanto, são adequados para longas distâncias e ambientes ruidosos (magnéticos).
- A alimentação CA é mais fácil e menos cara de fornecer ao equipamento.
- Os sinais CA são muito comuns em muitos dispositivos de automação existentes.

Os módulos de entrada possuem um circuito eletrônico com a função de converter o sinal elétrico externo em uma informação digital que será disponibilizada para a CPU no momento do processo de leitura das entradas. Esse tipo de circuito pode ser visto na Figura 1.14. Se o sinal elétrico externo for de 0 Volts, o circuito eletrônico coloca o valor binário “0” na memória correspondente na memória digital das entradas. Se o sinal elétrico externo for de 24 Volts, o circuito eletrônico coloca o valor binário “1” na memória correspondente na memória digital das entradas. Essa memória digital das entradas vai ser lida e o conteúdo vai ser transferida para a Tabela de Imagem das Entradas quando da execução do programa de aplicação.

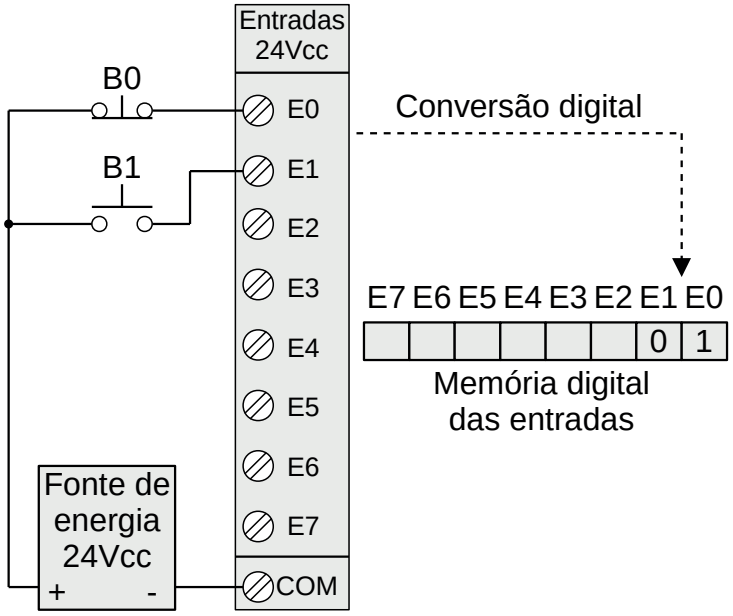

Figura 1.14 – Conversão do dispositivo físico para lógica binária.

As entradas do CLP devem converter uma variedade de níveis lógicos para os níveis lógicos de 5 Vcc usados no barramento de dados. Isso pode ser feito com circuitos semelhantes aos mostrados abaixo. Basicamente, os circuitos condicionam a entrada para acionar um opto acoplador. Isso isola eletricamente o circuito elétrico externo do circuito interno. Outros componentes do circuito são usados para proteger contra polaridade de tensão em excesso ou reversa.

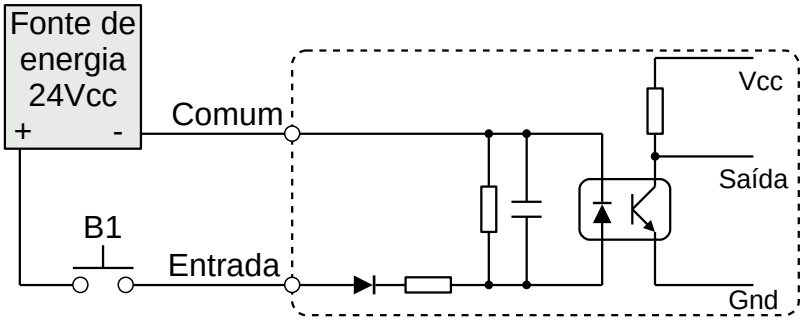

Figura 1.15 – Configuração típica de uma entrada tipo consumidora (negativo comum).

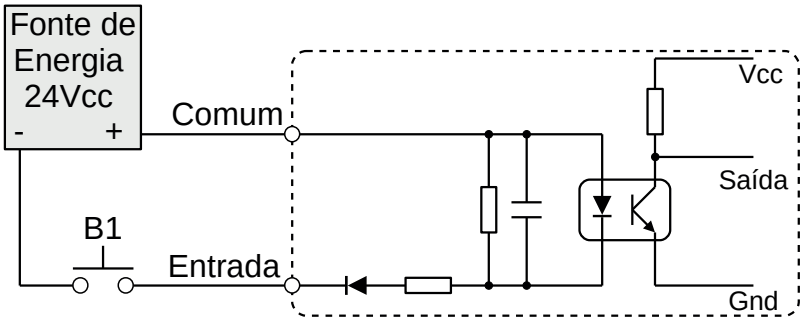

Figura 1.16 – Configuração típica de uma entrada tipo fornecedora (positivo comum).

Saídas

Como ocorre com os módulos de entrada, os módulos de saída raramente fornecem energia, mas atuam como interruptores. Fontes de alimentação externas são conectadas ao cartão de saída e o cartão liga ou desliga cada saída. As tensões de saída típicas estão listadas abaixo e ordenadas aproximadamente por popularidade.

120 Vca
24 Vcc
12-48 Vca
12-48 Vcc
5Vcc (TTL)
230 Vca

Esses cartões normalmente têm de 8 a 16 saídas do mesmo tipo e podem ser adquiridos com diferentes classificações de corrente. Uma escolha comum ao comprar cartões de saída são relés, transistores ou triacs. Os relés são os dispositivos de saída mais flexíveis. Eles são capazes de comutar saídas CA e CC. Mas, eles são mais lentos (cerca de 10 ms de comutação é comum), são mais volumosos, custam mais e se desgastam após milhões de ciclos. As saídas de relé são frequentemente chamadas de contatos secos. Os transistores são limitados às saídas DC e os Triacs são limitados às saídas AC. As saídas do transistor e triac são chamadas de saídas comutadas.

Contatos secos – um relé separado é dedicado a cada saída. Isso permite tensões mistas (AC ou DC e níveis de tensão até o máximo), bem como saídas isoladas para proteger outras saídas e o CLP. Os tempos de resposta costumam ser maiores que 10 ms. Este método é o menos sensível a variações e picos de tensão.

Saídas comutadas – uma tensão é fornecida ao cartão CLP e o cartão comuta para diferentes saídas usando circuitos de estado sólido (transistores, triacs, etc.) Os triacs são adequados para dispositivos AC que requerem menos de 1A. As saídas do transistor usam transistores NPN ou PNP de até 1A normalmente. Seu tempo de resposta é bem inferior a 1 ms.

É necessário cuidado ao construir um sistema com saídas CA e CC. Se CA for acidentalmente conectado a uma saída de transistor CC, ele ficará ligado apenas na metade positiva do ciclo e parecerá estar funcionando com uma tensão reduzida. Se a CC estiver conectada a uma saída CA triac, ela ligará e parecerá funcionar, mas você não poderá desligá-la sem desligar todo o CLP.

Lembrando: Um transistor é um dispositivo semicondutor que pode atuar como uma válvula ajustável. Quando desligado, ele bloqueará o fluxo de corrente em ambas as direções. Enquanto ligado, ele permitirá o fluxo de corrente apenas em uma direção. Normalmente ocorre uma perda de alguns volts no transistor. Um triac é como dois SCRs (ou transistores imaginários) conectados entre si para que a corrente possa fluir em ambas as direções, o que é bom para a corrente CA. Uma diferença importante para um triac é que se ele foi ligado para que a corrente flua e depois desligado, ele não desligará até que a corrente pare de fluir. Isso é bom para a corrente CA porque a corrente para e reverte a cada 1/2 ciclo, mas isso não acontece com a corrente CC e, portanto, o triac permanecerá ligado.

Um grande problema com as saídas são as fontes de alimentação misturadas. É uma boa prática é isolar todas as fontes de alimentação e manter seus comuns separados, mas isso nem sempre é viável. Alguns módulos de saída, como relés, permitem que cada saída tenha seu próprio comum. Outras placas de saída exigem que várias ou todas as saídas em cada placa compartilhem o mesmo comum. Cada placa de saída será isolada do resto, então cada comum terá que ser conectado. É comum para iniciantes conectar apenas o comum a um cartão e esquecer os outros cartões - então, apenas um cartão parece funcionar!

A placa de saída mostrada na Figura 1.17 é um exemplo de placa de saída 24 Vcc que tem um comum compartilhado. Esse tipo de placa de saída normalmente usa transistores para as saídas.

Neste exemplo, as saídas são conectadas a um elemento luminoso de baixa corrente (lâmpada) e uma bobina de relé. Considere o circuito através da lâmpada, começando com a alimentação de 24 Vcc. Quando a saída 07 está ligada, a corrente pode fluir em 07 para o COM, completando o circuito e permitindo o acendimento da lâmpada. Se a saída estiver desligada, a corrente não pode fluir e a lâmpada não acenderá. A saída 03 para o relé é conectada de forma semelhante. Quando a saída 03 está ligada, a corrente fluirá pela bobina do relé para fechar os contatos e fornecer 127 Vca para o motor.

Esta placa poderia ter muitas tensões diferentes aplicadas de fontes diferentes, mas todas as fontes de alimentação precisariam de um único comum compartilhado. Os circuitos na Figura 1.17 tem a sequência de fonte de alimentação, dispositivo, placa CLP e fonte de alimentação. Isso requer que a placa de saída tenha um terminal "comum". Alguns esquemas de saída invertem o dispositivo e a placa CLP, substituindo assim o comum por uma entrada de tensão. O exemplo da Figura 1.17 é repetido na Figura 1.18 para uma placa de alimentação de tensão.

Neste exemplo, o terminal positivo da alimentação de 24 Vcc é conectado diretamente ao cartão de saída. Quando uma saída está ligada, a energia será fornecida a essa saída. Por exemplo, se a saída 07 estiver ligada, a tensão de alimentação será enviada para a lâmpada. A corrente fluirá através da lâmpada e de volta ao comum na fonte de alimentação. O funcionamento é muito semelhante para o relé que liga o motor. Com este tipo de placa de saída, apenas uma fonte de alimentação pode ser usada.

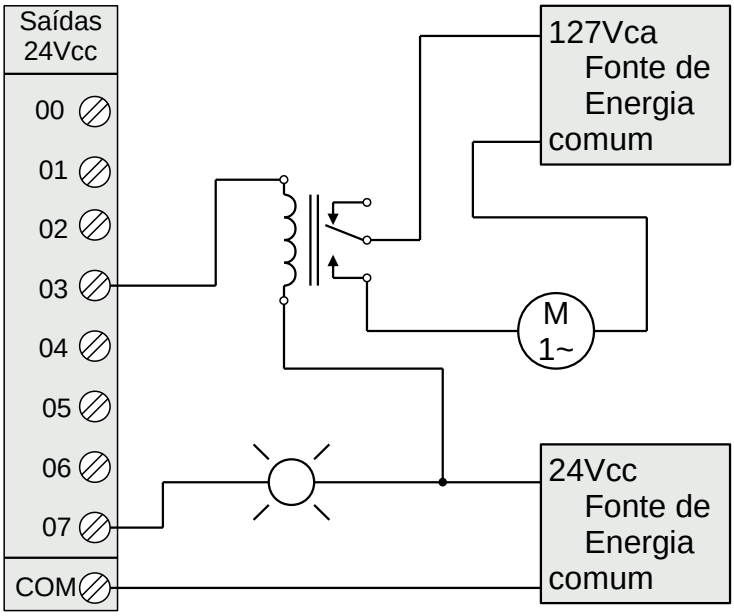

Figura 1.17 – Um cartão de saída CC (consumidor).

Também podemos usar saídas de relé para alternar as saídas. Existem dois tipos de placas com saídas em relé: terminal N.A. e terminal COM separados (dois parafusos por saída) e terminais COM interligados (um parafuso por saída). Com uma placa de saída de relés com terminais separados, pode-se ligar fontes de tensão CC e CA diretamente na placa.

Os circuitos eletrônicos mais comuns dos módulos de saída pode ser visto nas figuras 1.19 e 1.20.

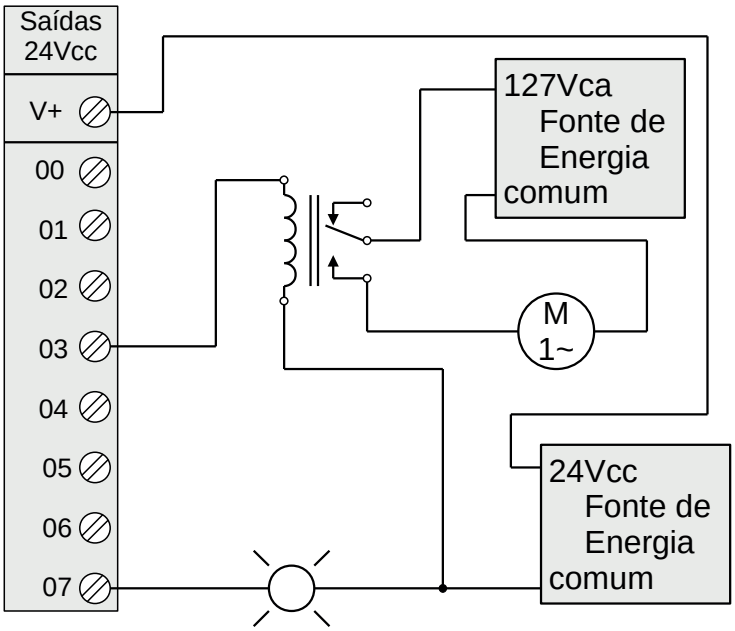

Figura 1.18 – Um cartão de saída CC (fornecedor).

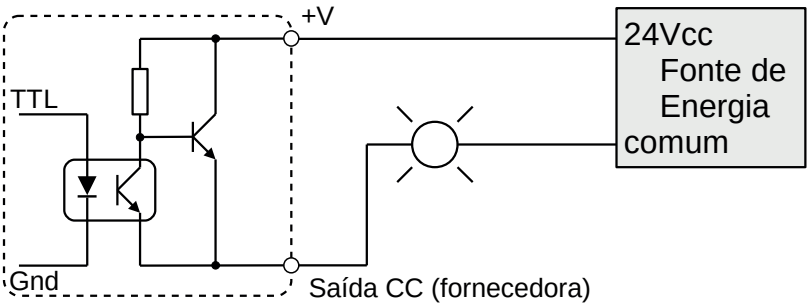

Figura 1.19 – Configuração típica de uma saída a transistor.

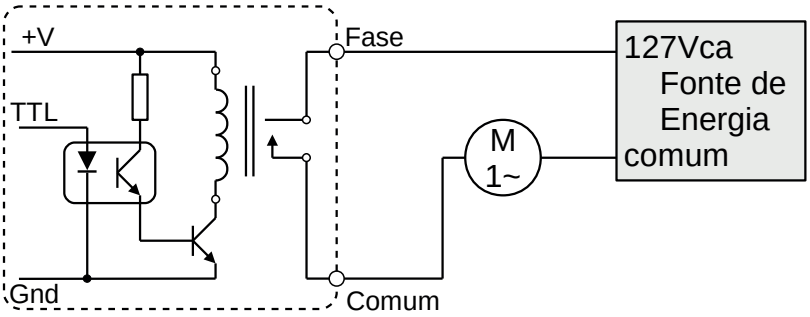

Figura 1.20 – Configuração típica de uma saída a relé.

Relés

Embora os relés raramente sejam usados para lógica de controle, eles ainda são essenciais para chavear grandes cargas de energia. Algumas terminologias importantes para relés são fornecidas abaixo.

Contator – Relés especiais para comutação de grandes cargas de corrente.

Motor Starter – Basicamente um contator em série com um relé de sobrecarga para desligar quando muita corrente é consumida.

Supressão de arco – quando qualquer relé é aberto ou fechado, um arco pula. Isso se torna um grande problema com relés grandes. Em relés comutando CA, este problema pode ser superado abrindo o relé quando a tensão vai para zero (ao cruzar entre negativo e positivo). Ao alternar as cargas CC, este problema pode ser minimizado soprando gás pressurizado durante a abertura para suprimir a formação do arco.

Bobinas CA – Se uma bobina normal for acionada por energia CA, os contatos vibrarão abrindo e fechando na frequência da energia CA. Este problema é superado pelos fabricantes de relés adicionando um polo de proteção à construção interna do relé.

A consideração mais importante ao selecionar relés, ou saídas de relé em um CLP, é a corrente e a tensão nominais. Se a tensão nominal for excedida, os contatos se desgastarão prematuramente ou, se a tensão for muito alta, pode ocorrer um incêndio. A corrente nominal é a corrente máxima que deve ser usada. Quando isso for excedido, o dispositivo ficará muito quente e falhará mais cedo. Os valores nominais são normalmente fornecidos para CA e CC, embora as classificações CC sejam mais baixas do que CA. Se as cargas reais usadas estiverem abaixo dos valores nominais, os relés devem funcionar bem indefinidamente. Se os valores forem excedidos um pouco, a vida útil do relé será reduzida de acordo. Exceder os valores significativamente pode causar falha imediata e danos permanentes. Observe que os relés também podem incluir classificações mínimas que também devem ser observadas para garantir a operação adequada e longa vida útil.

Tensão nominal – A tensão de operação sugerida para a bobina. Níveis mais baixos podem resultar em falha de operação, tensões acima encurtam a vida útil.

Corrente nominal – A corrente máxima antes que ocorra dano de contato (solda ou derrete).

1.3 – Estrutura funcional do CLP

1.3.1 – Ciclo de execução do CLP

O controlador lógico programável executa várias atividades desde o momento em que é ligado. Essas atividades são:

1. O CLP é energizado.
2. A CPU inicializa do hardware, verificando os parâmetros de memória retentiva. De modo geral, o conteúdo desta área é preservado, são mantidos os dados que havia quando o CLP foi desligado (desde que haja bateria de backup ou uso de memória flash), e a área de memória não retentiva é inicializada com zero.
3. A CPU verifica a configuração atual dos pontos de entrada e saída (relação de posição no rack versus módulo) e faz o seu mapeamento. Se houver diferença entre a configuração atual e a anterior, que existia quando o CLP foi desligado, a CPU pode indicar Erro Fatal, se tal opção estive habilitada.
4. A CPU realiza a leitura de todos os pontos de entrada, armazenando-os na Tabela de Imagem das Entradas.
5. A CPU verifica se há algum dispositivo periférico requisitando comunicação por meio de porta serial e faz o atendimento a ele.
6. A CPU atualiza a área especial de memória reservada ao relógio e calendário e aos relés especiais (como as marcas de diagnóstico). Esta área pode ser acessada pelo usuário apenas como leitura.
7. A CPU verifica o modo de operação selecionado: RUN (execução do programa de aplicação) ou PGM/STOP (programação / parada) e determina o fluxo de operação. Se estiver no modo PGM salta para o item 10. Se estive no modo STOP o controlador bloqueia todas as saídas mas continua a execução do programa.

8. A CPU executa o programa de aplicação desenvolvido pelo usuário para o controle desejado.
9. A CPU escreve o conteúdo da Tabela de Imagem das Saídas, gerada durante a execução do programa de aplicação, nos pontos de saída correspondentes.
10. A CPU realiza todos os diagnósticos do sistema, determinados pelo fabricante, além de outras tarefas, tais como: cálculo do tempo de varredura do programa, atualização de relés especiais e reset do "Watchdog Timer".
11. Se nenhum erro for diagnosticado, a CPU retorna ao início do ciclo, item 4, realizando nova leitura dos pontos de entrada.
12. A CPU faz a indicação do erro detectado por meio de marcadores (relés especiais) ou variáveis internas e LEDs externos.
13. Se o erro detectado não for fatal (não coloca em risco a CPU e/ou o equipamento), a CPU retorna ao início do ciclo, item 4, realizando nova leitura dos pontos de entrada.
14. A CPU é forçada (colocada) ao modo de programação, parando a execução do programa de aplicação e mantendo todas as saídas desligadas.

O ciclo normal, após a inicialização e sem erros, pode ser resumido como:

1. Atualização das entradas (leitura das entradas).
2. Execução do programa de aplicação (programa do usuário).
3. Atualização das saídas (escrita nas saídas).
4. Realização de diagnósticos.

Atualização das entradas

Os módulos de entradas podem ter ou não uma memória digital, conforme mostrado na Figura 1.14.

Se o módulo de entrada não possui esta memória, os dados são transferidos para a Tabela de Imagem das Entradas pela leitura bit a bit. Se a memória digital está presente, a função de gravação é desativada e a função de leitura é ativada, e os dados são transferidos para a Tabela de Imagem das Entradas. Um exemplo de mapeamento de memória de entrada é mostrado na figura 1.21.

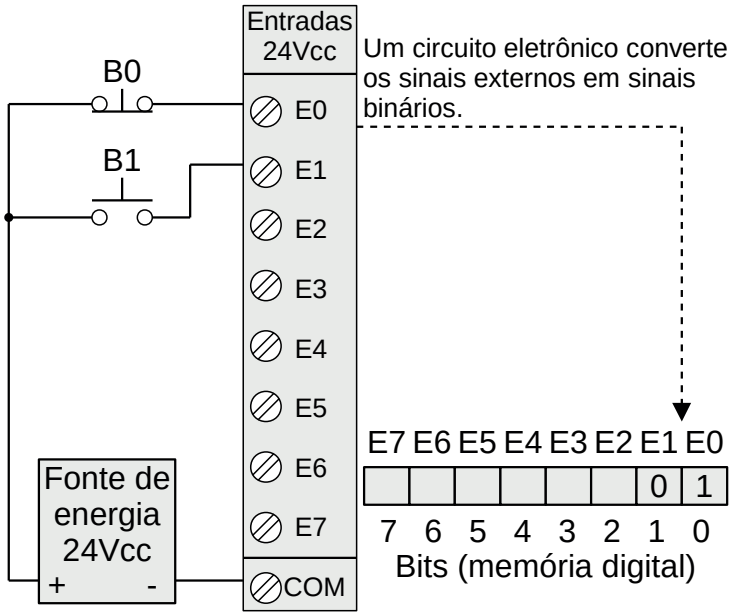

Figura 1.21 – Memória digital do módulo de entrada.

Execução do programa de aplicação

Após a leitura das entradas, a Tabela de Imagem das Entradas fica assim:

	E7	E6	E5	E4	E3	E2	E1	E0
	0	0	0	0	0	0	0	1
Bit	7	6	5	4	3	2	1	0

O programa de aplicação é executado, fazendo as leituras desta memória e não da memória que está no módulo de entrada. Se uma entrada tiver uma mudança durante a execução do programa de aplicação, será considerado o estado anterior da entrada. Um exemplo de programa de aplicação é mostrado na figura 1.22.

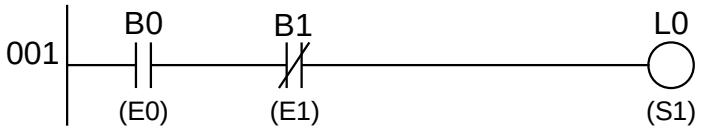

Figura 1.22 – Programa de aplicação.

Após a execução do programa de aplicação, a Tabela de Imagens das Saídas fica assim:

	S7	S6	S5	S4	S3	S2	S1	S0
	0	0	0	0	0	0	0	1
Bit	7	6	5	4	3	2	1	0

Importante: todas as linhas do programa de aplicação são executadas considerando a Tabela de Imagens das Entradas e fazendo as alterações na Tabela de Imagens das Saídas. Deve-se ter em mente que uma saída física não será alterada até que todo o programa de aplicação seja executado. Assim, deve-se verificar se uma determinada linha do programa altera um bit da Tabela de Imagens das Saídas e deve-se evitar que outra linha também faça a mesma alteração.

Atualização das saídas

Após o programa de aplicação terminar, o conteúdo da Tabela de Imagens das Saídas é copiado para a memória do módulo de saída correspondente. Um exemplo de mapeamento de memória de saída é mostrado na figura 1.23.

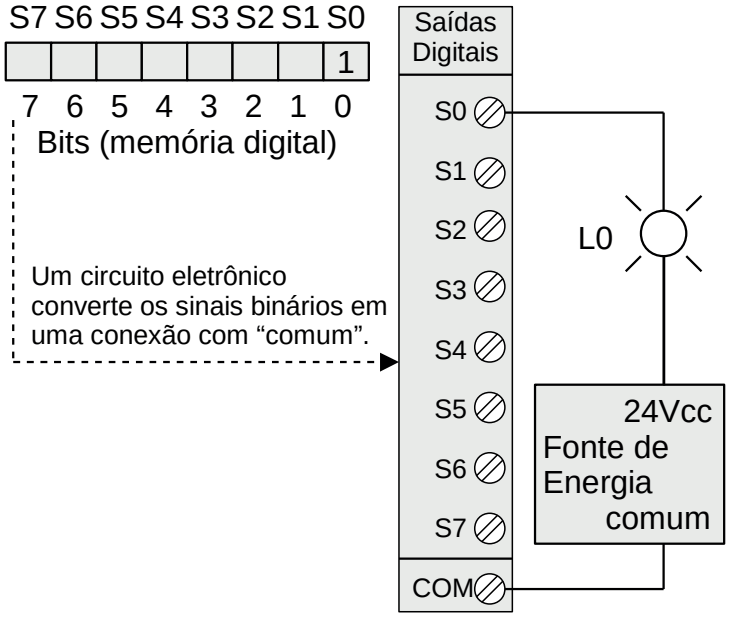

Figura 1.23 – Memória digital do módulo de saída.

Realização de diagnósticos

A CPU realiza todos os diagnósticos do sistema, alguns mais importantes:

- Cálculo do "Scan Time".
- Atualizar relés especiais correspondentes.
- Controle do "Watchdog Timer".

"Scan Time": tempo consumido pela CPU para realizar todas as tarefas em um ciclo, desde o início (Atualização das Entradas) até o término do ciclo (Atualização das Saídas).

"Watchdog Timer": armazena o tempo máximo permitido para execução de cada ciclo (definido pelo usuário).

- Se este tempo for excedido (Erro Fatal): a CPU é forçada ao modo de Programação e todas as saídas são desligadas.
- Caso contrário: o valor do "Scan Time" é armazenado em variável apropriada para realização de estatísticas como tempo máximo e tempo mínimo.
- "Scan Time" e "Watchdog Timer" são reinicializados em cada ciclo.
- Todos os erros (Fatais ou Não Fatais) são indicados por marcadores.
- Em alguns casos, o CLP possui LEDs externos (parte frontal CPU e Módulos de E/S).
- Algumas CPUs dispõem também de variável para armazenamento do Código de Erro ocorrido durante a execução do último ciclo.

Consideração sobre "Scan Time"

Fatores que influenciam diretamente sobre o "Scan Time":

- Quantidade de módulos e pontos de entrada ("Atualização das entradas").
- Conexão de Dispositivo(s) Periférico(s) ("Atendimento a Serviço Periférico").

- Tamanho do Programa de Aplicação e Tipo das Instruções utilizadas (“Execução do Programa de Aplicação”).
- Quantidade de módulos e pontos de saída (“Atualização das Saídas”).

O “Scan Time” influencia:

- Tempo de Resposta de Entrada/Saída.

Fatores de Influência no Tempo de Resposta de Entrada/Saída:

- O ponto (segmento) do ciclo que houve alteração do Ponto de Entrada.
- Tempo de resposta do módulo de entrada (0→1 e 1→0).
- Tempo de resposta do módulo de saída (0→1 e 1→0).

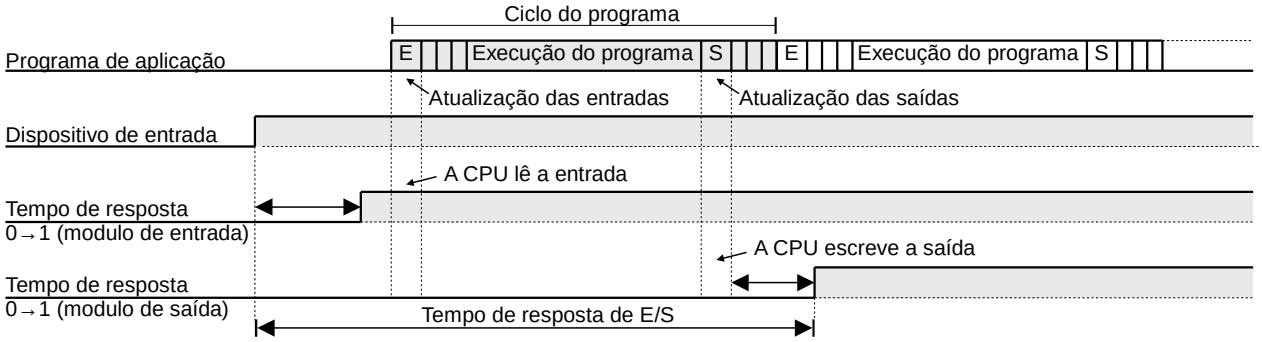

Figura 1.24 – Tempo de resposta de E/S mínimo.

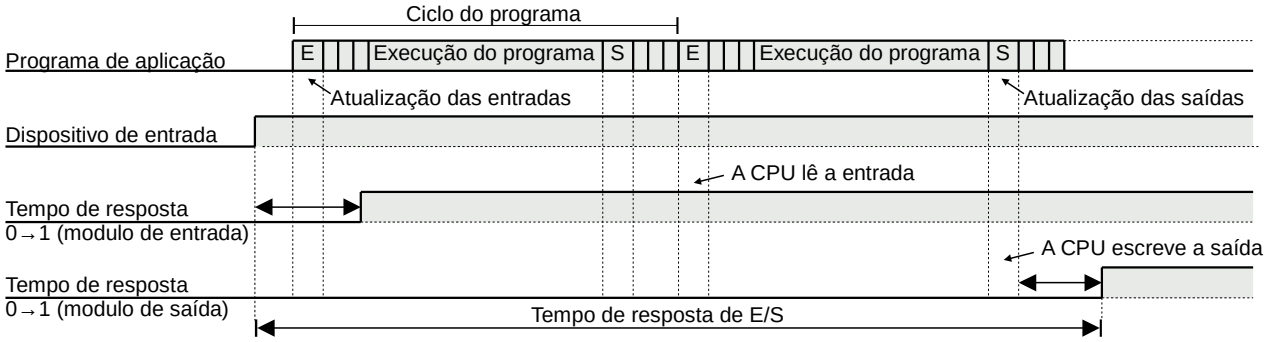

Figura 1.25 – Tempo de resposta de E/S máximo.

Instruções imediatas: acessam imediatamente os pontos de E/S no momento que são executadas. Normalmente usadas em aplicações de tempo de resposta críticos, onde é necessário um tempo menor que o tempo de ciclo do CLP. As Instruções Imediatas aumentam o “Scan Time “da CPU pois os módulos de E/S são acessados a cada execução de uma Instrução Imediata.

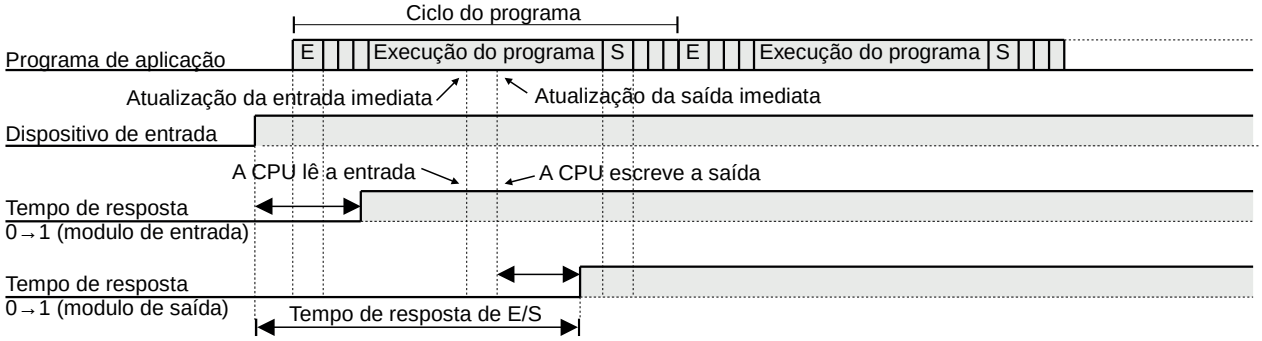

Figura 1.26 – Tempo de resposta para Instrução Imediata.

Tentativas para melhora do Tempo de Resposta de Entrada/Saída:

- Instruções com tempo de execução menor: diminui o “Scan Time”.
- Módulo de E/S com tempo de resposta menor: mais rápido.

1.4 – Linguagem Ladder

1.4.1 – Introdução

Um programa em Ladder permite que o controlador programável teste e modifique os dados por meio de símbolos gráficos padronizados. Esses símbolos são dispostos em redes de maneira semelhante a uma “linha” de um diagrama de lógica de relé. As redes Ladder são delimitadas à esquerda e à direita por trilhos de energia.

Trilhos de energia

A rede Ladder deve ser delimitada à esquerda por uma linha vertical conhecida como barramento de força esquerdo e à direita por uma linha vertical conhecida como barramento de força direito. O barramento de alimentação à direita pode ser explícito ou implícito.

Elementos e estados das ligações

Os elementos das ligações podem ser horizontais ou verticais. O estado do elemento da ligação deve obrigatoriamente ser denotado “Ligado (ON)” ou “Desligado (OFF)”, correspondendo aos valores booleanos literais 1 ou 0, respectivamente. O termo estado da ligação deve ser sinônimo do termo fluxo de potência.

O estado do trilho esquerdo deve ser considerado “Ligado” em todos os momentos. Nenhum estado é definido para o trilho direito. Um elemento de ligação horizontal deve ser indicado por uma linha horizontal. Um elemento de ligação horizontal transmite o estado do elemento imediatamente à esquerda para o elemento imediatamente à direita.

O elemento de ligação vertical deve consistir em uma linha vertical que se cruza com um ou mais elementos de ligação horizontal em cada lado. O estado da ligação vertical deve representar a lógica OU inclusive dos estados “Ligado” das ligações horizontais em seu lado esquerdo, ou seja, o estado da ligação vertical deve ser:

- “Desligado” se os estados de todas as ligações horizontais anexados à sua esquerda estiverem “desligadas”;
- “Ligado” se o estado de um ou mais das ligações horizontais anexados à sua esquerda estiverem “ligadas”.

O estado da ligação vertical deve ser copiado para todos as ligações horizontais anexados à sua direita. O estado da ligação vertical não deve ser copiado para nenhuma das ligações horizontais anexados à sua esquerda.

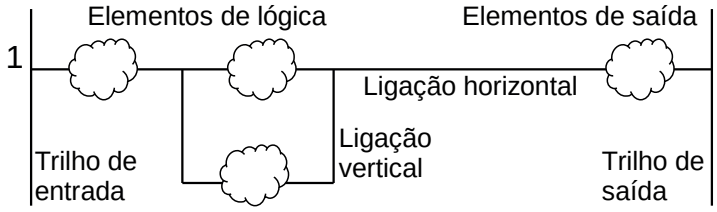

Figura 1.27 – Elementos de ligação da linguagem Ladder.

1.4.2 – Contatos

Um contato é um elemento que confere um estado à ligação horizontal em seu lado direito que é igual à lógica "E" booleana do estado da ligação horizontal em seu lado esquerdo com uma função apropriada de uma entrada, saída ou variável de memória Booleana associada. Um contato não modifica o valor da variável booleana associada. Os símbolos de contato padrão são fornecidos na Figura 1.28. As variáveis associadas podem ser de entradas digitais ou de uma saída digital ou de um bloco de função ou a uma memória interna. A exceção é o contato para leitura imediata que é aplicada apenas às entradas digitais.

Contato normalmente aberto

O estado da ligação à esquerda é copiado para a ligação à direita se o estado da variável booleana associada for "Ligado". Caso contrário, o estado da ligação à direita é "Desligado".

Contato normalmente fechado

O estado da ligação à esquerda é copiado para a ligação à direita se o estado da variável booleana associada for "Desligado". Caso contrário, o estado da ligação à direita é "Desligado".

Contato de detecção de transição positiva

O estado da ligação à direita é "Ligado" de uma avaliação deste elemento para a próxima quando uma transição da variável associada de "Desligado" para "Ligado" é detectada ao mesmo tempo que o estado da ligação à esquerda é "Ligado". O estado da ligação à direita deve estar "Desligado" em todos os outros momentos. Esse contato também é chamado de borda de subida e a simbologia pode ser alterada por alguns fabricantes para uma seta apontando para cima (↑) no lugar da letra (P).

Contato de detecção de transição negativa

O estado da ligação à direita é "Ligado" de uma avaliação deste elemento para a próxima quando uma transição da variável associada de "Ligado" para "Desligado" é detectada ao mesmo tempo que o estado da ligação à esquerda é "Ligado". O estado da ligação à direita deve estar "Desligado" em todos os outros momentos. Esse contato também é chamado de borda de descida e a simbologia pode ser alterada por alguns fabricantes para uma seta apontando para baixo (↓) no lugar da letra (N).

Contato para leitura imediata

Essa instrução força a CPU a uma leitura imediata da entrada especificada. O comportamento é semelhante ao do contato normalmente aberto. Essa instrução pode não estar disponível em alguns CLPs. Alguns fabricantes incorporam a função de leitura imediata para contato normalmente aberto e normalmente fechado.

Outros tipos de contato

Alguns fabricantes incorporam contatos nos seus CLPs para executarem tarefas de comparação como: igual (=), maior (>), menor (<), maior ou igual (>=), menor ou igual (<=) e diferente (≠).

Contatos especiais

Praticamente todos os fabricantes incorporam contatos especiais nos seus CLPs para executarem tarefas específicas como: primeiro ciclo, osciladores, marcadores de horários, indicadores de erros e outros.

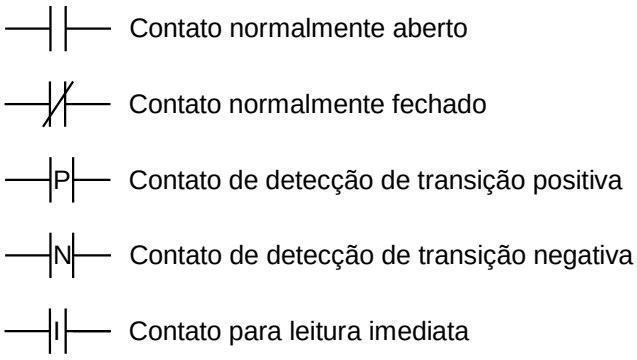

Figura 1.28 – Contatos da linguagem Ladder.

1.4.3 – Bobinas

Uma bobina copia o estado da ligação à sua esquerda para a ligação à direita sem modificação e armazena uma função apropriada do estado ou transição da ligação à esquerda na variável booleana associada. Os símbolos de bobina padrão são fornecidos na Figura 1.29.

Bobina direta

O estado da ligação à esquerda é copiado para a variável booleana associada e para a ligação à direita.

Bobina invertida

O estado da ligação à esquerda é copiado para a ligação à direita. O inverso do estado da ligação à esquerda é copiado para a variável booleana associada, ou seja, se o estado da ligação à esquerda é “Desligado”, então o estado da variável associada é “Ligado” e vice-versa.

Bobina Set (trava)

A variável booleana associada é configurada para o estado “Ligado” quando da ligação à esquerda está no estado “Ligado”, e permanece configurada até ser “destravada” por uma bobina “Reset”.

Bobina Reset (destravar)

A variável booleana associada é redefinida para o estado “Desligado” quando da ligação à esquerda está no estado “Ligado”, e permanece redefinida até ser “travada” por uma bobina “Set”.

Bobina de detecção de transição positiva

O estado da variável booleana associada é “Ligado” de uma avaliação deste elemento para a próxima quando uma transição da ligação à esquerda de “Desligado” para “Ligado” é detectada. O estado da ligação à esquerda é sempre copiado para a ligação à direita.

Bobina de detecção de transição negativa

O estado da variável booleana associada é “Ligado” de uma avaliação deste elemento para a próxima quando uma transição da ligação à esquerda de “Ligado” para “Desligado” é detectada. O estado da ligação à esquerda é sempre copiado para a ligação à direita.

Bobinas especiais

Alguns fabricantes incorporam bobinas em seus CLPs para executarem tarefas especiais como: pulso único (monoestável), oscilador, retentivo, alternativo (flip-flop T) e horários.

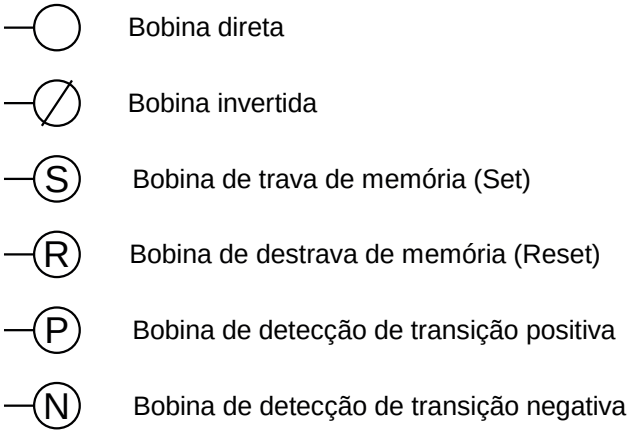

Figura 1.29 – Bobinas da linguagem Ladder.

1.4.4 – Blocos de função

Para fins de linguagens de programação de controlador programável, um bloco de função é uma unidade de organização de programa que, quando executada, produz um ou mais valores. Podem ser criadas várias instâncias nomeadas (cópias) de um bloco de funções. Cada instância deve ter um identificador associado (o nome da instância) e uma estrutura de dados contendo sua saída e variáveis internas e, dependendo da implementação, valores ou referências a suas variáveis de entrada. Todos os valores das variáveis de saída e as variáveis internas necessárias desta estrutura de dados devem persistir de uma execução do bloco de função para a próxima; portanto, a invocação de um bloco de função com os mesmos argumentos (variáveis de entrada) nem sempre precisa produzir os mesmos valores de saída.

Apenas as variáveis de entrada e saída devem ser acessíveis fora de uma instância de um bloco de função, ou seja, as variáveis internas do bloco de função devem ser ocultadas do usuário do bloco de função. Alguns fabricantes disponibilizam o acesso a essas variáveis internas diretamente ou indiretamente.

Temporizadores

Os temporizadores são contadores de unidades de tempo selecionáveis. Um temporizador pode apresentar opções para seleção do tipo de operação, origem do valor máximo de contagem, origem de parada de contagem e “reset” da contagem. Os símbolos padronizados para temporizadores e os diagrama de tempo podem ser vistos na figura 1.30 a 1.32. Nas figuras de simbologia, os códigos significam:

- IN (Input) – Sinal lógico digital para ativação do temporizador.
- PT (Preset Time) – Valor inteiro (normalmente de 16 bits) para especificar o limite de contagem de tempo.
- Q (Output) – Sinal lógico digital de saída do temporizador.
- ET (Elapsed Time) – Tempo decorrido desde a ativação do temporizador.

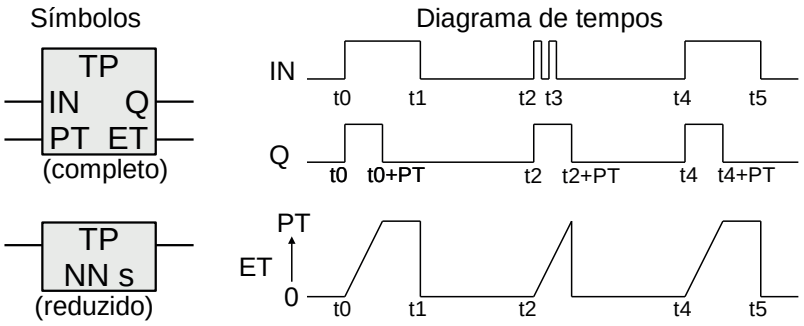

Figura 1.30 – Temporizador para pulso (TP) da linguagem Ladder.

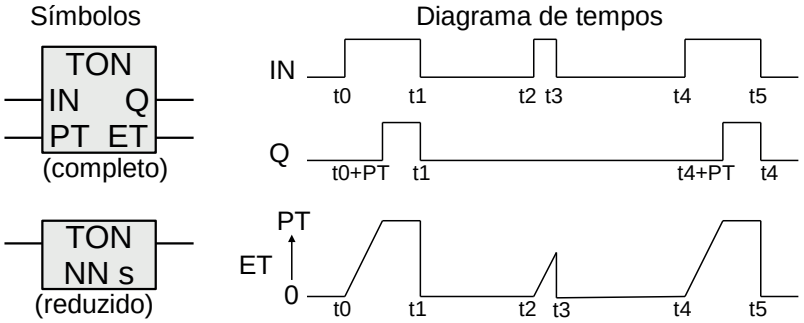

Figura 1.31 – Temporizador para ligar (TON) da linguagem Ladder.

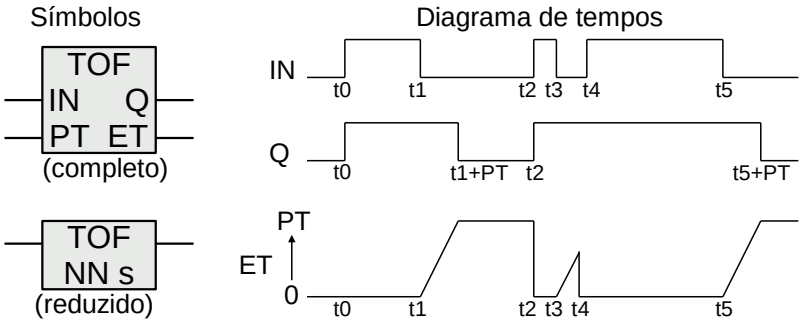

Figura 1.32 – Temporizador para desligar (TOF) da linguagem Ladder.

Contadores

Os contadores contam e acumulam a quantidade de eventos. Um contador pode apresentar opções para seleção do tipo de operação, origem do valor máximo e mínimo de contagem, origem de parada de contagem e "reset" da contagem. Os símbolos padronizados para contadores e os programas associados podem ser vistos na figura 1.33 a 1.35. Nas figuras de simbologia, os códigos significam:

- CU (Counter Up) – Sinal lógico digital para contagem progressiva.
- CD (Counter Down) – Sinal lógico digital para contagem regressiva.
- R (Reset) – Sinal lógico digital para zerar a contagem.
- LD (Load) – Sinal lógico digital para carregar o acumulador com o valor de PV.
- PV (Preset Value) – Valor inteiro (normalmente de 16 bits) para especificar o limite de contagem de eventos.
- Q (Output) – Sinal lógico digital de saída do contador.
- CV (Current Value) – Valor de contagem no acumulador.

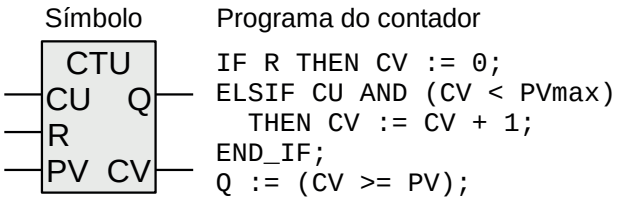

Figura 1.33 – Contador progressivo (CTU) da linguagem Ladder.

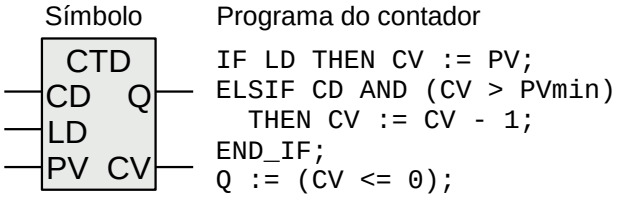

Figura 1.34 – Contador regressivo (CTD) da linguagem Ladder.

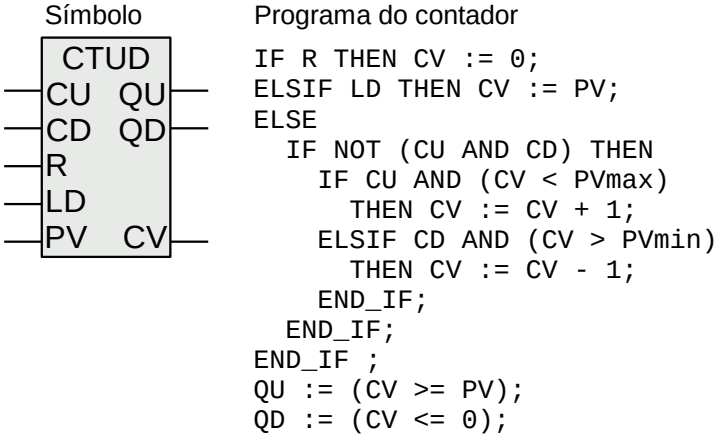

Figura 1.35 – Contador progressivo e regressivo (CTUD) da linguagem Ladder.

Outros blocos

Os fabricantes de CLPs incorporam vários blocos funcionais para facilitar a programação e aplicação do CLP nos mais diversos segmentos da indústria. Os blocos de função são divididos em áreas como: comunicação, apresentação (IHM) e controle analógico.

Em controle analógico é comum encontrar blocos para:

- Entrada analógica.
- Saída analógica.
- Registrador de dados (normalmente de 16 bits).
- Ganho de entrada analógica.
- Adição e subtração.
- Multiplicação e divisão.
- Multiplexador de dados.
- Comparador.
- Gerador de rampa – patamar.
- Controle ligado – desligado.
- Controle PID.
- Saída PWM.

- Funções matemáticas avançadas (raiz quadrada, trigonometria, logaritmos, potência).
- Funções de conversão de dados.
- Função para deslocamento de bits.

1.5 – Elementos de lógica em linguagem Ladder

1.5.1 – Introdução

Os elementos gráficos da linguagem Ladder são colocados em uma rede de linhas e colunas. Cada fabricante possui limitações de parâmetros normalmente associados às capacidades de memória disponíveis.

Como os contatos estão associados a elementos de entrada, variáveis internas e elementos de saída, raramente existe limitação de quantitativo de uso.

Como as bobinas de saída dependem dos módulos de saída digital, a sua quantidade no programa depende da quantidade disponível nos módulos de saída.

Normalmente as bobinas internas (relés auxiliares) tem a quantidade limitada pelo fabricante do CLP. Outros fabricantes limitam a quantidade de bobinas internas pela quantidade de memória disponível.

Os blocos de função são limitados pelo fabricante do CLP, sendo que alguns blocos, como temporizadores e contadores, podem estar disponibilizados em grande número, e outros blocos, como PID, podem ser limitados a uma pequena quantidade.

A estrutura da grade de linhas e colunas também é limitado pelo fabricante do CLP. Alguns fabricantes podem alocar apenas 3 colunas para contatos e 1 coluna para bobinas, e outros fabricantes podem alocar até 1024 colunas para contatos, blocos e bobinas. Existem CLPs com apenas uma única página de programa e outros com paginação múltipla para alocação do programa.

1.5.2 – Elementos de lógica

Associação direta

A lógica mais elementar em linguagem Ladder pode ser vista na figura 1.36, onde um elemento de contato é associado diretamente a um elemento de bobina. O contato pode ter origem em uma entrada física ou uma variável interna ou de uma bobina de saída. Uma das aplicações desta técnica é a ativação de uma saída digital baseada em um evento de uma memória interna ou a inversão de um valor interno.

Na linha 1, se o contato A1 for verdadeiro então a bobina Q1 é verdadeiro. Para que a bobina Q1 seja falsa é necessário que o contato A1 também seja falso.

Na linha 2, se o contato B1 for falso então a bobina Q2 é verdadeiro. Para que a bobina Q2 seja falsa é necessário que o contato B1 seja verdadeiro.

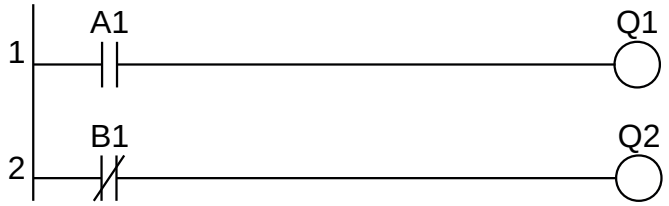

Figura 1.36 – Associação direta em linguagem Ladder.

Lógica booleana “INV”

A inversão ou negação de um elemento de lógica booleana em linguagem Ladder é representada simplesmente pelo contato ou bobina com uma barra inclinada (/) no contato ou bobina, não sendo necessário nenhuma estrutura mais complexa.

Lógica booleana “E”

A lógica booleana “E” é implementada em linguagem Ladder por dois ou mais contatos na mesma linha e em série, conforme mostrado na figura 1.37. Essa é uma das funções mais usuais em programação na linguagem Ladder para o modo estruturado de programação.

Na linha 1, se o contato A1 for verdadeiro e o contato A2 for verdadeiro então a bobina Q1 é verdadeiro. Para que a bobina Q1 seja falsa é necessário que apenas um dos contatos, A1 e A2, também seja falso.

Na linha 2, se o contato B1 for falso e o contato B2 for falso então a bobina Q2 é verdadeiro. Para que a bobina Q2 seja falsa é necessário que apenas um dos contatos, B1 ou B2, seja verdadeiro.

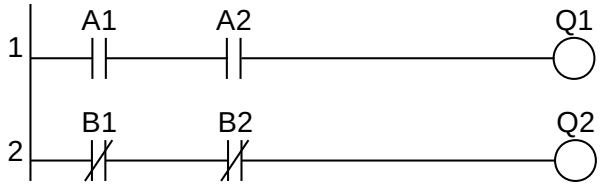

Figura 1.37 – Lógica booleana “E” em linguagem Ladder.

Lógica booleana “OU”

A lógica booleana “E” é implementada em linguagem Ladder por dois ou mais contatos na mesma linha e em paralelo, conforme mostrado na figura 1.38. Essa função é utilizada quando qualquer um de dois ou mais eventos podem ativar uma variável interna ou uma saída.

Na linha 1, se o contato A1 for verdadeiro ou o contato A2 for verdadeiro então a bobina Q1 é verdadeiro. Para que a bobina Q1 seja falsa é necessário que os dois contatos, A1 e A2, também sejam falsos.

Na linha 2, se o contato B1 for falso ou o contato B2 for falso então a bobina Q2 é verdadeiro. Para que a bobina Q2 seja falsa é necessário que os dois contatos, B1 e B2, sejam verdadeiros.

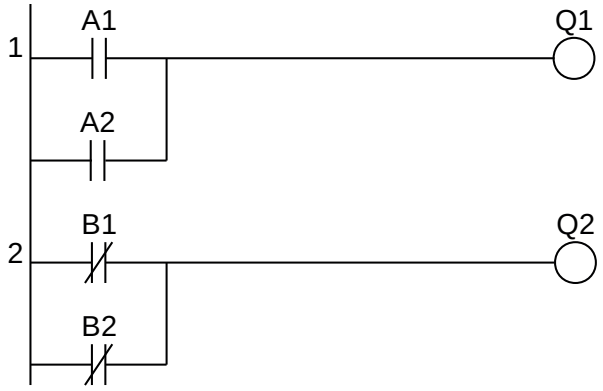

Figura 1.38 – Lógica booleana “OU” em linguagem Ladder.

Lógica booleana “OU-Exclusivo”

A lógica booleana “OU-Exlusivo” é implementada em linguagem Ladder por dois contatos em série e em paralelo na mesma linha, conforme mostrado na figura 1.39. Essa função é utilizada quando dois eventos não podem ocorrer em simultâneo.

Na linha 1, se o contato A1 for verdadeiro e o contato A2 for falso ou se o contato A1 for falso e o contato A2 for verdadeiro, então a bobina Q1 é verdadeira. Para que a bobina Q1 seja falsa é necessário que os dois contatos, A1 e A2, também sejam ambos falsos ou ambos verdeiros.

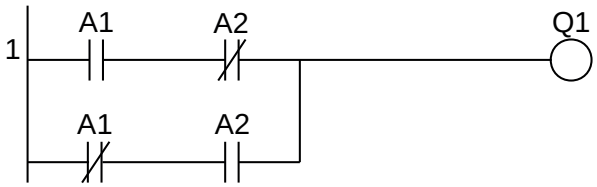

Figura 1.39 – Lógica booleana “OU-Exclusivo” em linguagem Ladder.

1.5.3 – Elementos de memória

Latch RS

O elemento de memória mais simples é o Latch RS. A posição de memória, que pode ser uma variável interna ou uma saída digital, responde imediatamente às entradas de ativação de memória (torna a memória verdadeira) e desativação de memória (torna a memória falsa).

O programa em linguagem Ladder da figura 1.40 pode ser entendido como:

- Ativação da memória: se o contato S for verdadeiro ou o contato Q for verdadeiro e o contato R for falso então a bobina Q é verdadeira.
- Desativação da memória: se o contato R for verdadeiro, e independente das condições dos contatos S ou Q, então a bobina Q é verdadeira.

Esse programa utiliza a técnica de selo do contato de ativação da memória. Observe que se S e R estiverem verdadeiros em simultâneo, a saída Q permanece em falso.

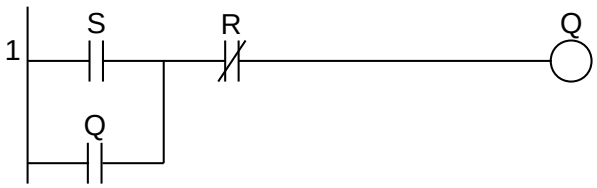

Figura 1.40 – Latch RS (versão 1) em linguagem Ladder.

O programa em linguagem Ladder da figura 1.41 pode ser entendido como:

- Ativação da memória: se o contato S for verdadeiro então a bobina Q é verdadeira e permanece verdadeiro.
- Desativação da memória: se o contato R for verdadeiro então a bobina Q é falsa e permanece falsa.

Esse programa utiliza a técnica de bobina especial de ativação e desativação de memória. Observe que se S e R estiverem verdadeiros em simultâneo, a saída Q permanece em falso, devido que a bobina de Reset está após a bobina de Set. Se as linhas 1 e 2 forem invertidas, a bobina de Set fica após a bobina de Reset e então se S e R estiverem verdadeiros em simultâneo, a saída Q permanece em verdadeiro.

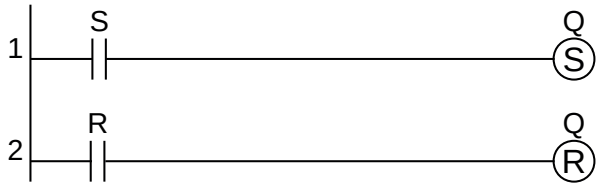

Figura 1.41 – Latch RS (versão 2) em linguagem Ladder.

O programa em linguagem Ladder da figura 1.42 apresenta um comportamento mais realístico de um Latch ou Flip-flop tipo RS. As linhas 2 e 3 emulam uma porta lógica "NOU" interligadas. A linha 1 foi introduzida para evitar a bobina "C2" seja verdadeira no primeiro ciclo do programa.

As linhas 4 e 5 implementam a saída direta e complementar, respectivamente. Uma entrada de ativação ("Enable") pode ser facilmente realizada por inserção de um contato em série com o contato "R" e em série com o contato "S".

O funcionamento deste circuito é:

a) No primeiro ciclo do programa, a variável "PTD" é falso e isso evita que a bobina "C2" seja verdadeira. Com "C2" falso então, pela linha 2, a bobina "C1" torna-se verdadeira. No próximo ciclo do programa, a variável "PTD" torna-se verdadeira. A saída "Q" é falsa e a saída "Q" é verdadeira.

b) Se o contato S for verdadeiro e o contato R for falso, a variável "C2" torna-se verdadeiro, independente da condição de "C1" e "PTD" na linha 3. No ciclo seguinte do programa, "C1" torna-se falso devido a "C2" ser verdadeiro. A variável "C2" permanecerá verdadeira, independente da condição de "S". A saída "Q" é verdadeira e a saída "Q" é falsa.

c) Se o contato R for verdadeiro e o contato S for falso, a variável "C1" torna-se verdadeiro, independente da condição de "C2" na linha 2. No ciclo seguinte do programa, "C2" torna-se falso devido a "C1" ser verdadeiro. A variável "C1" permanecerá verdadeira, independente da condição de "R". A saída "Q" é falsa e a saída "Q" é verdadeira.

d) Se o contato R for verdadeiro a variável "C1" torna-se verdadeiro, na linha 2. E se o contato S for verdadeiro a variável "C2" torna-se verdadeiro, na linha 3. A saída "Q" é falsa e a saída "Q" é falsa.

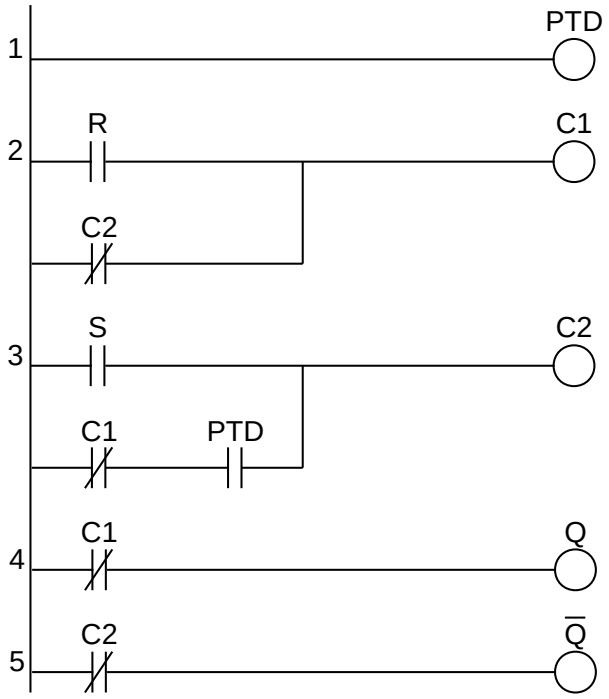

Figura 1.42 – Flip-Flop RS em linguagem Ladder.

Flip-Flop D

O elemento de memória mais complexo é o Flip-flop tipo D. A posição de memória, que pode ser uma variável interna ou uma saída digital, será igual à entrada se um sinal de ativação estiver em verdadeiro. Assim, se o contato de dados (“D”) for verdadeiro e o contato de ativação (“E”) for verdadeiro é feito a ativação de memória (torna a memória verdadeira) e se o contato de dados (“D”) for falso e o contato de ativação (“E”) for verdadeiro é feito a desativação de memória (torna a memória falsa). Se o contato de ativação (“E”) for falso a memória não é alterada. Um Flip-flop tipo D é construído a partir de um Latch RS e pode ser visto na figura 1.43.

O funcionamento deste circuito é:

e) No primeiro ciclo do programa, a variável “PTD” é falso e isso evita que a bobina “C2” seja verdadeira. Com “C2” falso então, pela linha 2, a bobina “C1” torna-se verdadeira. No próximo ciclo do programa, a variável “PTD” torna-se verdadeira. A saída “Q” é falsa e a saída “Q” é verdadeira.

f) Se o contato “E” for verdadeiro e o contato “D” for verdadeiro, a variável “C2” torna-se verdadeiro, independente da condição de “C1” e “PTD” na linha 3. No ciclo seguinte do programa, “C1” torna-se falso devido a “C2” ser verdadeiro. A variável “C2” permanecerá verdadeira enquanto a condição de “D” não se alterar. A saída “Q” é verdadeira e a saída “Q” é falsa.

g) Se o contato “E” for verdadeiro e o contato “D” for falso, a variável “C1” torna-se verdadeiro, independente da condição de “C2” na linha 2. No ciclo seguinte do programa, “C2” torna-se falso devido a “C1” ser verdadeiro. A variável “C1” permanecerá verdadeira enquanto a condição de “D” não se alterar. A saída “Q” é falsa e a saída “Q” é verdadeira.

h) Se o contato “E” for falso e independente da condição do contato “D” não haverá alteração em “C1” e “C2” e portanto a saída “Q” e a saída “Q” não serão alteradas.

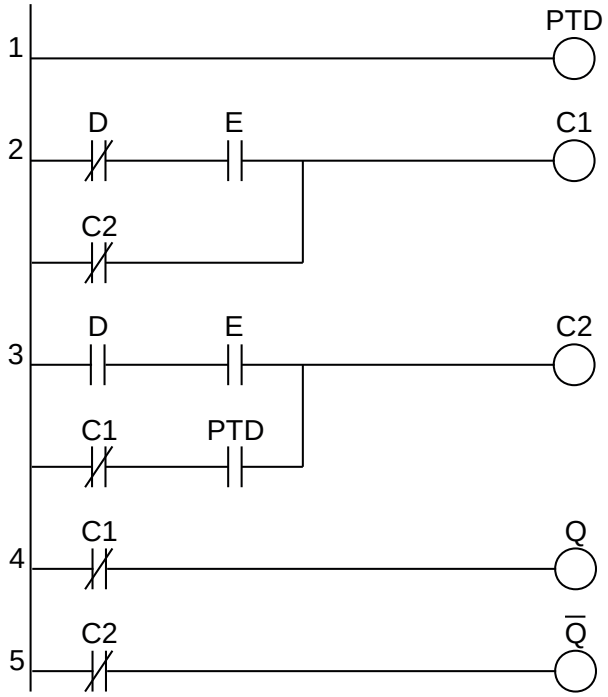

Figura 1.43 – Flip-Flop D em linguagem Ladder.

Flip-Flop JK

O flip-flop tipo JK pode ser implementado na linguagem Ladder conforme pode ser visto na figura 1.44. Esse tipo de flip-flop opera com duas entradas “J” e “K” que definem o modo de operação

e a entrada “CLK” para ativar o modo de operação. Observe que o contato “CLK” está na configuração “um pulso”, ou seja, no primeiro ciclo que “CLK” for verdadeiro, as linhas 2 e 3 estão habilitadas. Como a linha 2 é executada primeiro que a linha 3, a lógica “CLK.C2” é verdadeira. No próximo ciclo do programa esta lógica é falso. A linha 1 é um seletor de modo de operação. A linha 4 possui um “OU-Exclusivo” entre as saídas Q e C1.

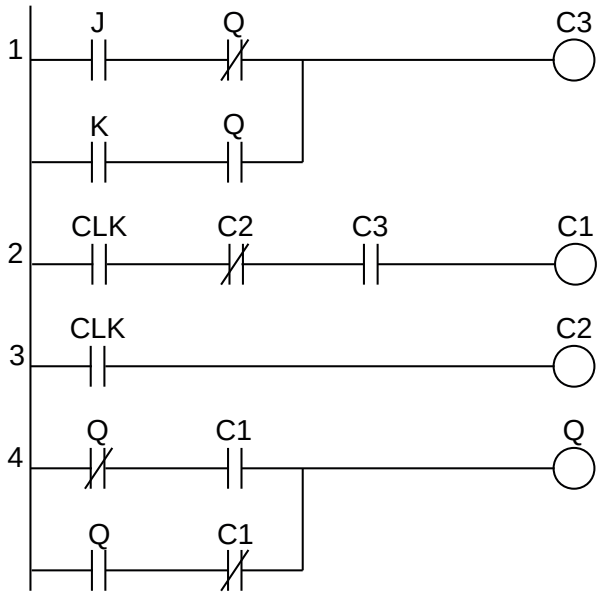

Figura 1.44 – Flip-Flop JK em linguagem Ladder.

Modo 1: “J” é falso e “K” é falso. Q não altera.

Linha 1: C3 é falso, independente da saída Q.

Linha 2: C1 é falso, independente do pulso de CLK.

Linha 4: Se Q é falso e C1 é falso então Q permanece falso. Se Q é verdadeiro e C1 é falso, a lógica Q.C1 é verdadeiro então Q permanece verdadeiro.

Modo 2: “J” é verdadeiro e “K” é falso. Q é verdadeiro.

Linha 1: C3 é verdadeiro se Q é falso e C3 é falso se Q é verdadeiro.

Linha 2: C1 é verdadeiro se C3 é verdadeiro e C1 é falso se C3 é falso.

Linha 4: Se Q é falso e C1 é verdadeiro então Q é verdadeiro. Se Q é verdadeiro e C1 é falso, a lógica Q.C1 é verdadeiro então Q permanece verdadeiro.

Modo 3: “J” é falso e “K” é verdadeiro. Q é falso.

Linha 1: C3 é verdadeiro se Q é verdadeiro e C3 é falso se Q é falso.

Linha 2: C1 é verdadeiro se C3 é verdadeiro e C1 é falso se C3 é falso.

Linha 4: Se Q é verdadeiro e C1 é verdadeiro então Q é falso. Se Q é falso e C1 é verdadeiro, a lógica Q.C1 é verdadeiro então Q permanece verdadeiro.

Modo 4: “J” é verdadeiro e “K” é verdadeiro. Q muda de condição.

Linha 1: C3 é verdadeiro, independente de Q.

Linha 2: C1 é verdadeiro durante o pulso de CLK.

Linha 4: Se Q é verdadeiro e C1 é verdadeiro então Q é falso. Se Q é falso e C1 é verdadeiro, a lógica Q.C1 é verdadeiro então Q é verdadeiro.

Flip-Flop T

O modo 4 do flip-flop JK implementa o flip-flop tipo T. Outro modo de representar o flip-flop tipo T pode ser visto na figura 1.45. O funcionamento básico de um flip-flop tipo T é: quando a variável "T" mudar de falso para verdadeiro a variável ME (mestre) muda de condição (se falso passa para verdadeiro e se verdadeiro passa para falso) e quando a variável "T" mudar de verdadeiro para falso a variável ES (escravo) muda de condição (se falso passa para verdadeiro e se verdadeiro passa para falso).

O funcionamento do programa é:

a) A condição inicial é: T falso, ME falso e ES (Q também é falso).

b) A variável T muda de falso para verdadeiro, a lógica ((T+ME).(Q+T)) ((1+0).(1+0) = 1) na linha 1 é verdadeira, então ME muda de falso para verdadeiro. Na linha 2, a lógica ((T+ES).(ME+T)) ((0+0).(1+1) = 0) é falsa e ES mantém falso. A saída Q também é falso.

c) A variável T muda de verdadeiro para falso, a lógica ((T+ME).(Q+T)) (0+1).(1+1) = 1) na linha 1 é verdadeira, então ME continua verdadeiro. Na linha 2, a lógica ((T+ES).(ME+T)) ((1+1).(1+0) = 1) é verdadeira e ES muda de falso para verdadeiro. A saída Q também é verdadeiro.

d) A variável T muda de falso para verdadeiro, a lógica ((T+ME).(Q+T)) ((1+0).(0+0) = 0) na linha 1 é falso, então ME muda de verdadeiro para falso. Na linha 2, a lógica ((T+ES).(ME+T)) ((0+1).(0+1) = 1) é verdadeiro e ES mantém verdadeiro. A saída Q também é verdadeiro.

e) A variável T muda de verdadeiro para falso, a lógica ((T+ME).(Q+T)) ((0+0).(1+1) = 0) na linha 1 é falso, então ME continua falso. Na linha 2, a lógica ((T+ES).(ME+T)) ((1+0).(0+0) = 0) é falso e ES muda de verdadeiro para falso. A saída Q também é falsa.

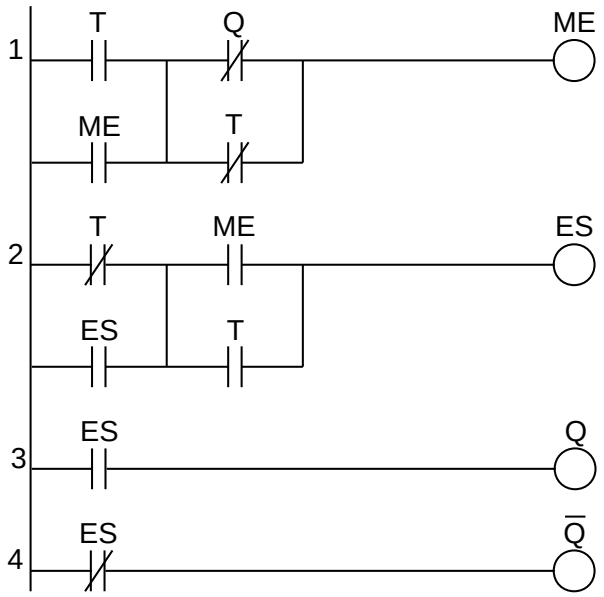

Figura 1.45 – Flip-Flop T em linguagem Ladder.

1.5.4 – Funções especiais

Algumas tarefas de programação em CLP exigem comandos especiais e alguns fabricantes não incorporam esses comandos no grupo de contatos especiais.

Primeiro ciclo (“first scan”)

Muitos programas, principalmente os sequenciais, precisam ser inicializados em algum ponto do programa. E para isso é necessário um pulso com duração de um ciclo do programa e funcionando apenas no primeiro ciclo. Esse contato deve ser verdadeiro ou falso no primeiro ciclo e depois mudar para falso ou verdadeiro nos ciclos seguintes. A figura 1.46 mostra essa configuração.

O funcionamento do programa é:

a) No primeiro ciclo do programa, na linha 1, FSt é falso, então FS é verdadeiro. Na linha 2, a lógica (FS+FSt) (1+0 = 1) é verdadeiro, então FSt é verdadeiro.

b) No próximo ciclo do programa, na linha 1, FSt é verdadeiro, então FS é falso. Na linha 2, a lógica (FS+FSt) (0+1 = 1) é verdadeiro, então FSt continua verdadeiro.

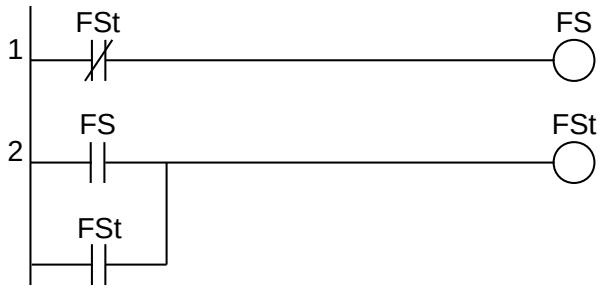

Figura 1.46 – Primeiro ciclo em linguagem Ladder.

O programa da figura 1.47 executa a mesma tarefa mas de um modo mais simplificado.

Figura 1.47 – Primeiro ciclo (simplificado) em linguagem Ladder.

Um pulso (“one shot”)

Alguns problemas ficam mais fáceis de solucionar quando um determinado evento ocorre por apenas um único ciclo do programa. Isso é conhecido como “um pulso” ou “pulso único”. Um modo de implementação desta função pode ser visto na figura 1.48.

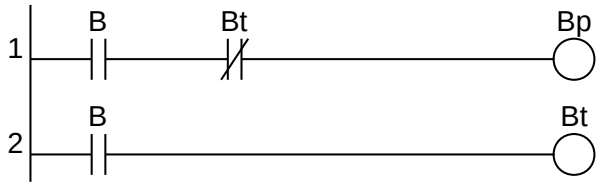

Figura 1.48 – Um pulso em linguagem Ladder.

O funcionamento do programa é:

a) O estado inicial é “Bp” falso e “Bt” falso.

b) Na linha 1, quando o contato “B” for verdadeiro, a lógica “B.Bt” (1.1 = 1) é verdadeiro, então a bobina “Bp” é verdadeiro. Na linha 2, quando o contato “B” for verdadeiro então a bobina “Bt” é verdadeiro.

c) No próximo ciclo do programa, se “B” continuar verdadeiro, a lógica “B.Bt” (1.0 = 0) é falso, então a bobina “Bp” é falso. Na linha 2, se “B” continuar verdadeiro então a bobina “Bt” é verdadeiro.

Na figura 1.49 é mostrado um programa melhorado para pulso único para borda positiva.

Quando B1 passar de falso para verdadeiro, C0 é verdadeiro no primeiro ciclo e depois é falso nos ciclos seguintes.

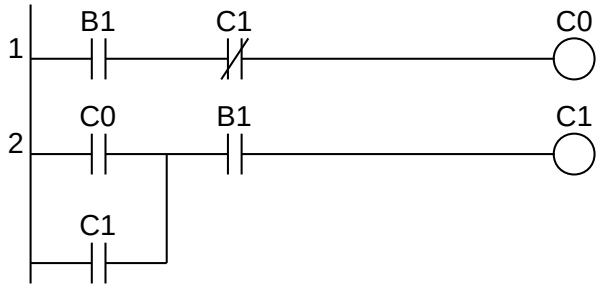

Figura 1.49 – Um pulso (borda positiva) em linguagem Ladder.

Na figura 1.50 é mostrado um programa melhorado para pulso único para borda negativa.

Quando B2 passar de verdadeiro para falso, C2 é verdadeiro no primeiro ciclo e depois é falso nos ciclos seguintes.

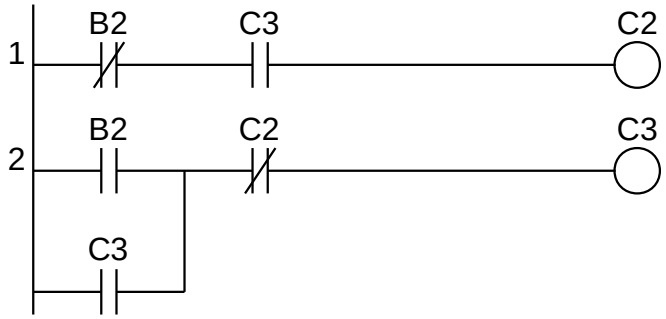

Figura 1.50 – Um pulso (borda negativa) em linguagem Ladder.

1.6 – Conclusão

Vantagens do controlador lógico programável (CLP)

O CLP tem muitas vantagens que o tornam um controlador de processo exclusivo. As vantagens estão sempre aumentando. Programadores, operadores e engenheiros preferem os CLPs a outros sistemas de controle, devido à sua simplicidade, segurança e confiabilidade.

O CLP é simples e flexível

Nenhum sistema de controle é fácil ou simples, mas os CLPs surgiram como comparativamente simples e fáceis para outros sistemas líderes de controle de processo. No CLP é fácil adicionar ou remover qualquer dispositivo final sem afetar todo o processo. Além disso, o CLP tem a flexibilidade de monitorar e corrigir o sistema a qualquer momento, mesmo na situação “On-line” ou “Off-line” da planta.

O CLP é distribuído

O sistema de controle alocado implica em muitos computadores e estações na rede ocupados em controle. O CLP é criado para tornar o sistema de controle alocado mais leve com os equipamentos de rede que estão inseparavelmente ligados a ele. O CLP é considerado o emissor e o receptor na gravação de arquivos de sistema de rede. Além disso, o CLP é capaz de trabalhar com as partes de hardware e software do computador.

O CLP é seguro

A segurança no CLP é excelente. É um dos primeiros programas de controle que possui a mais poderosa proteção contra riscos, perda de dados e danos aos usuários. Esses recursos tornaram o CLP um sistema de programa de controle sofisticado e útil. Comparando o CLP com o SCADA do lado da segurança, o CLP tem o melhor resultado.

O CLP é confiável

Tanto a confiabilidade quanto a segurança são importantes para cada sistema de controle. O CLP é um exemplo dos sistemas de controle que os possuem. O CLP oferece vários pontos de medidas de confiabilidade, começando com o próprio CLP e terminando com muitos de seus recursos. Por exemplo, ponteiros e conversão automática de tipo estão incluídos no CLP. Antes que o PLC termine de fazer e iniciar um programa, ele deve verificar se há algum erro que ele mesmo possa consertar ou questionar o usuário sobre isso.

O CLP é programável

A linguagem do controlador lógico programável (CLP) ajudou a desenvolver o sistema de controle e se tornou a linguagem mais útil. Os controladores lógicos programáveis se tornaram a linguagem usada pela maioria das empresas e fábricas. Eles fornecem e apoiam os engenheiros com o que precisam e mantêm os operadores protegidos de máquinas perigosas. O potencial dos controladores lógicos programáveis em sistemas de controle é quase ilimitado. Os controles lógicos programáveis alimentaram e deram suporte aos sistemas de controle de várias maneiras. A maioria dos computadores de processo utilizados em fábricas com humanos tem mais vantagens do que desvantagens como um sistema seguro ao lidar com processos complexos. Os controladores lógicos programáveis provaram ser uma ferramenta eficaz e útil em muitas indústrias no passado. Com suas muitas vantagens, eles continuarão a ser úteis também no futuro.

Capítulo 2
Programação por álgebra booleana

2.1 – Introdução

Este método de programação utiliza a álgebra booleana como base para montagem do programa do controlador lógico programável. Existem duas abordagens estratégicas para se conseguir as equações booleanas: usar a técnica de tabelas verdade e usar a técnica de frases lógicas.

O uso da tabela verdade é mais simples mas pode ser muito trabalhoso quando se tem muitas variáveis. Tem a vantagem de verificar todas as ocorrências relativas às entradas em relação a todas as saídas possíveis. A tabela verdade pode ser construída utilizando os min-termos ou os max-termos e ainda com situações não aplicáveis. A equação gerada, soma de produtos, pode ser minimizada utilizando métodos tradicionais de minimização como manipulação algébrica, mapa de Karnaugh ou algoritmo de Quine–McCluskey.

O uso de frases lógicas, ou sentenças lógicas, analisa todas as entradas e verifica quais são aptas a influenciar um evento numa determinada saída. Assim, uma saída é considerada verdadeira, se um conjunto de variáveis também é considerado verdadeiro. A frase lógica pode ser escrita em linguagem natural ou em matemática, utilizando os elementos de lógica (E, OU, Inversão). A frase lógica é então transformada em uma equação boolena e pode ser minimizada.

2.2 – Solução de problemas utilizando a tabela verdade

2.2.1 – Exemplo 1. Acionamento de um cilindro pneumático

Esse exemplo inicial mostra como um problema bem simples é solucionado com uma tabela verdade sem a necessidade de minimização da equação booleana.

Considere um problema de acionar um cilindro pneumático. Se o cilindro for de simples ação com retorno por mola, então é utilizado uma eletroválvula do tipo 3/2 vias. A montagem do circuito pneumático pode ser visto na Figura 2.1, onde o solenoide da eletroválvula recebe a etiqueta Y1.

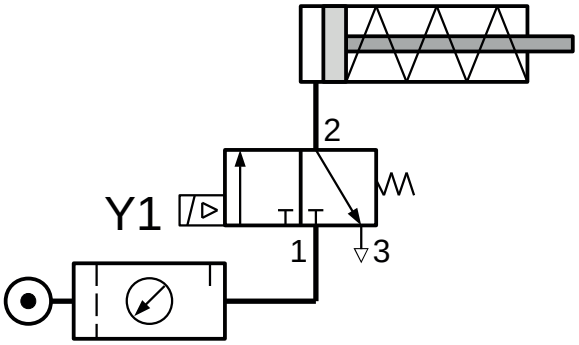

Figura 2.1 – Cilindro pneumático de simples ação e recuo por mola.

Observe que para o cilindro pneumático executar a ação de avanço, a eletroválvula tem que permanecer acionada durante todo o percurso do cilindro e, para tanto, o solenoide Y1 tem que estar acionado durante este tempo. Para o recuo do cilindro pneumático é necessário apenas cortar o sinal no

solenoide Y1, a válvula é colocada na posição inicial e a mola do cilindro pneumático faz o recuo expulsando o ar através da saída 3 da eletroválvula.

Se o cilindro pneumático deve ser acionado devido à ocorrência da combinação de 3 variáveis (A, B e C) então uma tabela verdade de 8 linhas deve ser montada. A lógica proposta é que apenas uma das entradas deve ser ativa e as outras entradas não ativas. Para representar que uma determinada entrada está ativa, utiliza-se o número "1" e para representar que uma determinada entrada não está ativa, utiliza-se o número "0". A Tabela 2.1 apresenta todas as 8 possibilidades das entradas A, B e C com relação à saída Y1.

C	B	A	Y1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Tabela 2.1 – Tabela verdade para Y1.

A equação booleana para esta tabela verdade é: $Y1 = A.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + \overline{A}.\overline{B}.\overline{C}$.

Esta equação pode ser simplificada utilizando manipulação algébrica booleana.

O diagrama Ladder para essa equação booleana não simplificada pode ser visto na Figura 2.2.

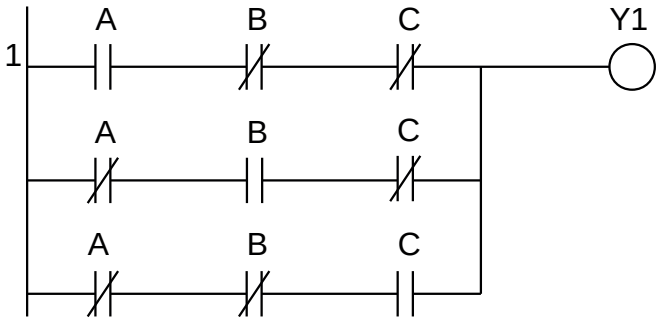

Figura 2.2 – Solução do exemplo 1 em linguagem Ladder.

Observe que as entradas A ou B ou C devem estar com, pelo menos, uma delas ativa para que a saída Y1 seja ativa. Se as entradas forem consideradas sensibilizadas por sensores momentâneos, a saída Y1 deve possuir uma memória de ativação e deve ser proposto uma outra regra para a desativação da saída Y1.

2.2.2 – Exemplo 2. Controle de uma bomba d'água

Esse exemplo mostra outro problema mais complexo onde a equação booleana encontrada pode ser minimizada por mapa de Karnaugh.

Descrição do problema

Uma bomba d'água do tipo centrífuga fornece água a um reservatório de grande dimensão, como está ilustrado na figura 2.3. A bomba é atuada pelo contator "S". São utilizados dois sensores lógicos, "L2m" e "L3m", para o nível da água respectivamente indicando se foram atingidos os dois metros e o três metros de altura. Os sensores são do tipo boia, ou seja, acima da altura para que foram calibrados passam a fornecer o valor lógico 1. Por outro lado existem duas válvulas solenoide "C" e "D", onde "C" escoa uma vazão maior que a "D", e que se destinam a retirar água do reservatório. Esses solenoides são controlados por outro processo. Cada válvula está equipada com um sensor lógico de abertura, "Cx" e "Dx" respectivamente.

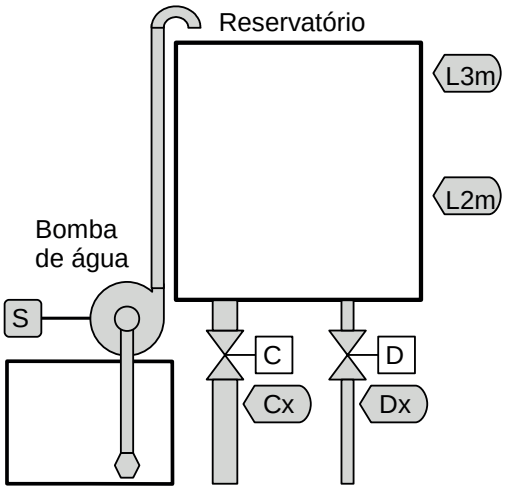

Figura 2.3 – Controle de uma bomba d'água.

Descrição operacional da máquina ou processo

Para prevenir que o nível da água não desça a níveis demasiado baixos, a bomba d'água deve ser acionada se alguma das seguintes situações ocorrer:

a) O nível da água desce abaixo dos 2 metros.

b) O nível da água está entre os 2 metros e os 3 metros e a válvula "C" está aberta.

c) O nível da água está acima dos 3 metros mas ambas as válvulas estão abertas.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 2.2.

Parafuso	Etiqueta	Descrição
I01	L3m	Sensor de nível tipo boia.
I02	L2m	Sensor de nível tipo boia.
I03	Cx	Fluxostato de líquido.
I04	Dx	Fluxostato de líquido.

Tabela 2.2 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 2.3.

Parafuso	Etiqueta	Descrição
Q01	S	Contator de motor da bomba d'água.

Tabela 2.3 – Descrição das saídas.

Solução por tabela verdade

Uma tabela verdade é montada para avaliar as condições de operação das saídas em relação às entradas. Esta tabela verdade pode ser vista na tabela 2.4.

S = 0 significa bomba d'água desligada.

S = 1 significa bomba d'água ligada.

S = – significa uma condição impossível (alarme).

P	L2m	L3m	Cx	Dx	S	Lógica
0	0	0	0	0	1	$\overline{L2m}.\overline{L3m}.\overline{Cx}.\overline{Dx}$
1	0	0	0	1	1	$\overline{L2m}.\overline{L3m}.\overline{Cx}.Dx$
2	0	0	1	0	1	$\overline{L2m}.\overline{L3m}.Cx.\overline{Dx}$
3	0	0	1	1	1	$\overline{L2m}.\overline{L3m}.Cx.Dx$
4	0	1	0	0	-	
5	0	1	0	1	-	
6	0	1	1	0	-	
7	0	1	1	1	-	
8	1	0	0	0	0	
9	1	0	0	1	0	
10	1	0	1	0	1	$L2m.\overline{L3m}.\overline{Cx}.Dx$
11	1	0	1	1	1	$L2m.\overline{L3m}.\overline{Cx}.\overline{Dx}$
12	1	1	0	0	0	
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	1	$L2m.L3m.Cx.Dx$

Tabela 2.4 – Tabela verdade para o exemplo 2.

A equação booleana cheia é a soma dos mintermos:

S = min(0) + min(1) + min(2) + min(3) + min(10) + min(11) + min(15)

Substituindo os mintermos "min(nn)" pelas entradas correspondentes:

$$S = \overline{L2m}.\overline{L3m}.\overline{Cx}.\overline{Dx} + \overline{L2m}.\overline{L3m}.\overline{Cx}.Dx + \overline{L2m}.\overline{L3m}.Cx.\overline{Dx} + \overline{L2m}.\overline{L3m}.Cx.Dx +$$
$$+ L2m.\overline{L3m}.\overline{Cx}.Dx + L2m.\overline{L3m}.\overline{Cx}.\overline{Dx} + L2m.L3m.Cx.Dx$$

A minimização dessa equação pode ser feita por mapa de Karnaugh. A tabela 2.5 mostra a posição de alocação de cada mintermo extraído da tabela verdade para 4 variáveis. A distribuição dos mintermos do exemplo 2 no mapa de Karnaugh pode ser visto na tabela 2.6.

Mintermos	$\overline{Cx}.\overline{Dx}$	$\overline{Cx}.Dx$	$Cx.Dx$	$Cx.\overline{Dx}$
$\overline{L2m}.\overline{L3m}$	**0**	**1**	**3**	**2**
$\overline{L2m}.L3m$	4	5	7	6
$L2m.L3m$	12	13	**15**	14
$L2m.\overline{L3m}$	8	9	**11**	**10**

Tabela 2.5 – Distribuição da posição dos mintermos num mapa de Karnaugh de 4 variáveis.

S	$\overline{Cx}.\overline{Dx}$	$\overline{Cx}.Dx$	$Cx.Dx$	$Cx.\overline{Dx}$
$\overline{L2m}.\overline{L3m}$	1	1	1	1
$\overline{L2m}.L3m$	0	0	0	0
$L2m.L3m$	0	0	1	0
$L2m.\overline{L3m}$	0	0	1	1

Tabela 2.6 – Alocação dos mintermos do exemplo 2 no mapa de Karnaugh.

Grupo 1: min(0) + min(1) + min(2) + min(3)

Minimizado: $\overline{L2m}.\overline{L3m}$

Grupo 2: min(2) + min(3) + min(10) + min(11)

Minimizado: $\overline{L3m}.Cx$

Grupo 3: min(11) + min(15)

Minimizado: $L2m.Cx.Dx$

A equação minimizada é: $S = \overline{L2m}.\overline{L3m} + \overline{L3m}.Cx + L2m.Cx.Dx$

Esta equação pode ser executada na linguagem Ladder e o programa pode ser visto na figura 2.4.

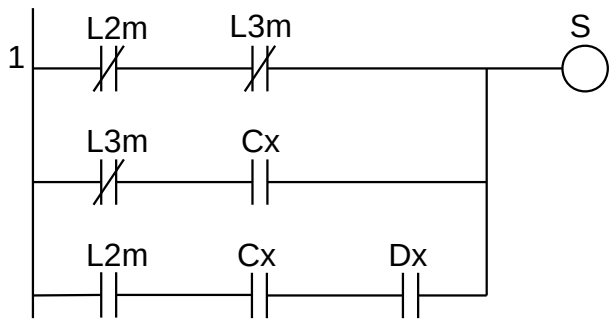

Figura 2.4 – Solução do exemplo 2 em linguagem Ladder.

2.2.3 – Exemplo 3. Estação de bombeamento com múltiplas entradas

Um exemplo de aplicação muito mais complexo é mostrado na Figura 2.5, que é um esquema de uma instalação de bombeamento de água. Neste exemplo é necessário a criação de variáveis internas para especificar uma condição da máquina. São elaboradas várias tabelas verdades para cada grupo de eventos.

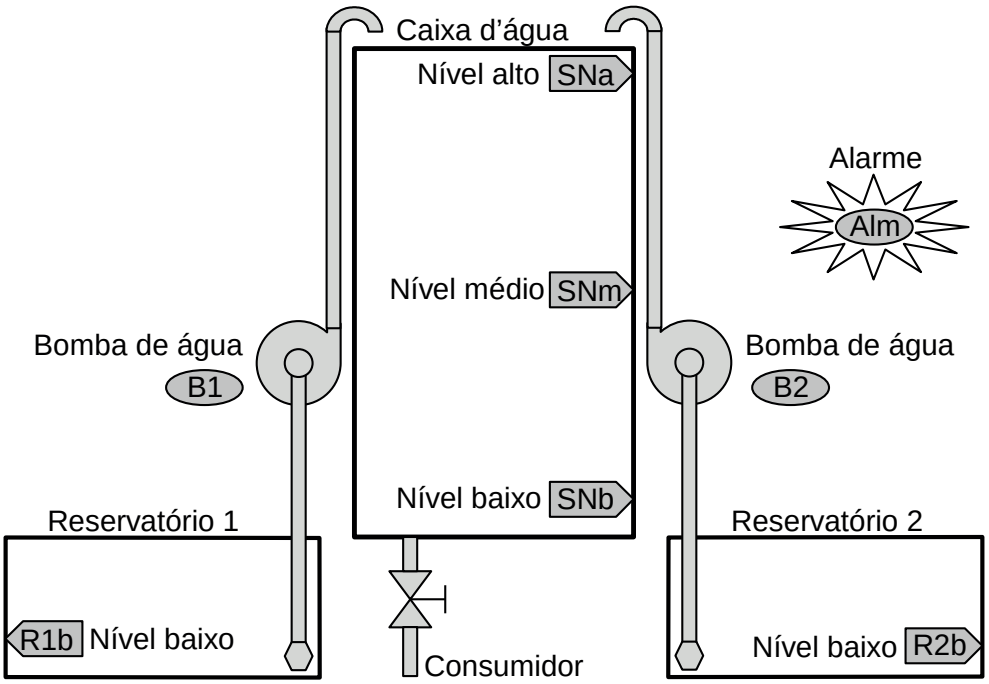

Figura 2.5 – Diagrama esquemático de uma instalação de bombeamento de água.

Descrição da máquina ou processo

Esta instalação de bombeamento de água é composta de uma caixa d'água em nível superior com 3 sensores de nível tipo boia, sendo etiquetadas como "SNa" para sensor de nível alto, "SNm" para sensor de nível médio e "SNb" para sensor de nível baixo. Existe uma saída de água para o consumidor desta caixa d'água. Existem dois reservatórios em níveis inferiores com sensores de nível tipo boia, sendo etiquetado como "R1b" para sensor de nível baixo no reservatório 1 e "R2b" para sensor de nível baixo no reservatório 2. Existem duas bombas d'água do tipo centrífuga etiquetadas como "B1" para bomba d'água 1 e "B2" para bomba d'água 2. Existe uma lâmpada de sinalização de alarme etiquetada como "Alm".

Os sensores de nível são ligados ao controlador utilizando os contatos NA (normalmente aberto). Os motores das bombas d'água são ligados em contatores de capacidade de corrente elétrica adequada. As bobinas dos contatores são em 127 Volts tensão alternada com corrente de 0,2 Amperes. A lâmpada sinalizadora é do tipo comum de baixa potência, 127 Volts tensão alternada com potência de 5 Watts.

Descrição operacional da máquina ou processo

As condições de funcionamento desta estação de bombeamento de água são as seguintes:

1. As bombas d'água somente poderão ser ligadas se tiver água nos reservatórios. Existe uma válvula direcional no início da tubulação dentro do reservatório que impede da água descer

pelo tubo de sucção, ou seja, a válvula permite o fluxo de água apenas de baixo para cima. A bomba e o cano devem estar permanentemente cheios de água para a bomba funcionar. Se entrar ar, a rotação da bomba não faz força de sucção suficiente para puxar a água do reservatório.

2. As bombas d'água são identificadas como "primária" e "secundária" através de algum dispositivo de seleção que pode ser externo ao controlador, como uma chave, ou interno ao controlador, como uma variável.
3. O acionamento das bombas deve seguir uma combinação dos sensores na caixa d'água:
   a) Se o nível de água na caixa d'água estiver em "baixo" ("SNb" verdadeiro), as duas bombas d'água devem ser acionadas.
   b) Se o nível de água na caixa d'água estiver em "médio" ("SNm" verdadeiro), somente uma bomba d'água deve ser acionada (bomba piloto, que pode ser a B1 ou B2).
   c) Se o nível de água na caixa d'água estiver em "alto" ("SNa" verdadeiro), a bomba d'água piloto deve ser desligada.
4. Se a caixa d'água estiver esvaziando, o acionamento das bombas deve ser:
   a) Se estiver esvaziando na metade alta, ou seja, entre os sensores "SNa" e "SNm", a bomba d'água piloto não deve ser acionada.
   b) Se estiver esvaziando na metade baixa, ou seja, entre os sensores "SNm" e "Snb", a bomba d'água secundária não deve ser acionada.
5. A lâmpada de sinalização de alarme deve acender de acordo com:
   a) Se os sensores de nível alto "SNa", médio "SNm" e baixo "SNb" não estiverem na sequência correta, a lâmpada de sinalização de alarme deve ficar acesa continuamente.
   b) Se os dois reservatórios d'água não tiverem água simultaneamente, a lâmpada de sinalização de alarme deve ficar acesa de modo piscante. Este alarme deve prevalecer sobre o alarme anterior.
6. Nenhuma das bombas d'água deve funcionar se houver a existência de qualquer um dos alarmes.
7. Se uma bomba d'água estiver no modo piloto e faltar água no reservatório correspondente, então a outra bomba torna-se piloto.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na Tabela 2.7.

Parafuso	Etiqueta	Descrição
I01	SNa	Sensor de nível alto na caixa d'água.
I02	SNm	Sensor de nível médio na caixa d'água.
I03	SNb	Sensor de nível baixo na caixa d'água.
I04	R1b	Sensor de nível baixo no reservatório 1.
I05	R2b	Sensor de nível baixo no reservatório 2.

Tabela 2.7 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na Tabela 2.8.

Parafuso	Etiqueta	Descrição
Q01	Alm	Lâmpada sinalizadora de alarme.
Q02	B1	Contator do motor da bomba d'água 1.
Q03	B2	Contator do motor da bomba d'água 2.

Tabela 2.8 – Descrição das saídas.

Observação: os nomes aplicados às colunas "parafuso" foram colocados de maneira genérica, sendo Ixx para entrada e Qxx para saídas. Cada fabricante de controlador para automação adota uma numeração própria. Não existe uma regra específica de identificação. Os nomes são encontrados nos pinos de conexão do controlador. Normalmente alguns projetistas de painel de controle utilizam borneiras de conexão com identificação própria e também com identificação no cabo de ligação. Neste caso, novas colunas podem ser adicionadas à tabela para melhorar a identificação. Um exemplo de conexão em painel pode ser visto na Figura 2.6, onde I1 é o texto da etiqueta do pino no CLP, X1 é o texto da etiqueta dos parafusos da borneira do CLP e D101 é o texto da etiqueta dos parafusos da borneira de dispositivos. FL101 é o texto da etiqueta do fio de ligação do parafuso do CLP ao parafuso da borneira do CLP. FI101 é o texto da etiqueta do fio de ligação do parafuso da borneira do CLP ao parafuso da borneira de dispositivos. FE101 é o texto da etiqueta do fio de ligação do parafuso da borneira de dispositivos ao dispositivo externo.

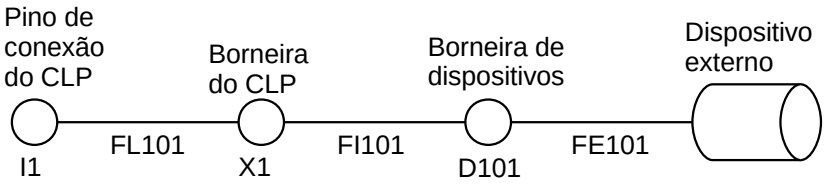

Figura 2.6 – Diagrama de conexões com borneiras.

Solução por tabela verdade

Vamos começar a solução do problema pelos alarmes. De acordo com a descrição de funcionamento 5a, se a sequência de ativação dos sensores da caixa d'água não estiver na sequência correta, deve ser acionado o alarme. Esta "sequência correta" é relativa ao preenchimento de água na caixa d'água. Assim, se a caixa d'água estiver enchendo, o sensor "SNb" deve ser acionado primeiro e depois "SNm" e depois "SNa". Uma tabela verdade pode ser montada e ela pode ser vista na Tabela 2.9. Observe que a saída foi chamada de "almc" como sendo uma variável interna que será utilizada para compor a saída "Alm". O símbolo "1" é utilizado para condição verdadeira e o símbolo "0" é utilizado para condição falsa.

D	SNb	SNm	SNa	almc
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Tabela 2.9 – Tabela verdade para "almc".

A equação booleana para esta tabela verdade é:

$$almc = \overline{SNb}.\overline{SNm}.SNa + \overline{SNb}.SNm.\overline{SNa} + \overline{SNb}.SNm.SNa + SNb.\overline{SNm}.SNa$$

Esta equação pode ser simplificada utilizando manipulação algébrica booleana ou mapa de Karnaugh.

Análise das linhas da tabela:

- Linha D0 – A caixa d'água está totalmente vazia ou o nível da água está abaixo do sensor "SNb".
- Linha D4 – Com o sensor "SNb" ativo, entende-se que o nível da água está entre os sensores "SNb" e "SNm".
- Linha D6 – Com os sensores "SNb" e "SNm" ativos, entende-se que o nível da água está entre os sensores "SNm" e "SNa".
- Linha D7 – Com todos os sensores ativos, entende-se que a caixa d'água está totalmente cheia.

Qualquer outra combinação deve ser entendido como defeito no sensor. O sensor pode ficar danificado em sempre falso ou sempre verdadeiro. Não é objetivo deste problema analisar e solucionar falhas em sensores. Isto será visto em outro capítulo deste livro.

As linhas D1, D2, D3 e D5 podem indicar falha nos sensores como sempre verdadeiro (símbolo 1) ou que os sensores anteriores estão em sempre falso (símbolo 0).

O diagrama em linguagem Ladder, não simplificado, pode ser visto na Figura 2.7.

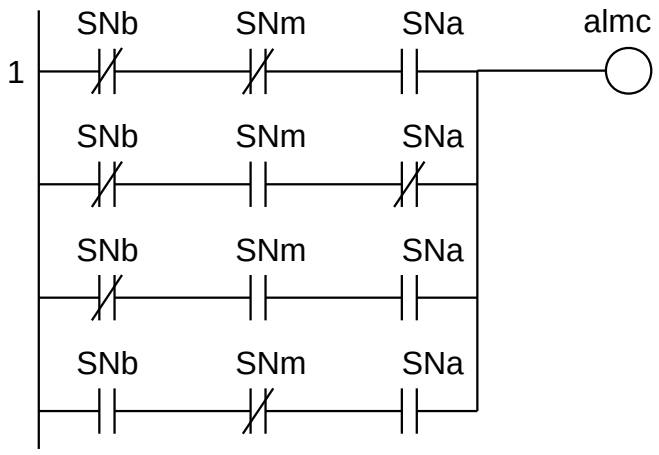

Figura 2.7 – Diagrama Ladder para a **Tabela 2.9**.

De acordo com a descrição de funcionamento 5b, o alarme piscante deve ser ativado se os dois reservatórios d'água não tiverem água simultaneamente. Uma tabela verdade pode ser montada e ela pode ser vista na Tabela 2.10. Observe que a saída foi chamada de "almr" como sendo uma variável interna que será utilizada para compor a saída "Alm". O símbolo "1" é utilizado para condição verdadeira e o símbolo "0" é utilizado para condição falsa.

D	R1b	R2b	almr
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

Tabela 2.10 – Tabela verdade para "almr".

A equação booleana para esta tabela verdade é: $almr = \overline{R1b}.\overline{R2b}$

Esta equação não precisa ser simplificada por manipulação algébrica booleana ou mapa de Karnaugh.

- A linha D0 indica a não existência de água nos dois reservatórios e portanto motivo de alarme.
- A linha D1 indica a existência de água no reservatório 2 mas não no reservatório 1.
- A linha D2 indica a existência de água no reservatório 1 mas não no reservatório 2.
- A linha D3 indica a existência de água nos dois reservatórios.

O diagrama em linguagem Ladder pode ser visto na Figura 2.8.

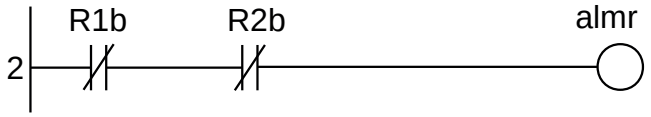

Figura 2.8 – Diagrama Ladder para a **Tabela 2.10**.

De acordo com a descrição de funcionamento 6, qualquer um dos alarmes deve bloquear o funcionamento das bombas d'água. Então uma nova variável é proposta para ser utilizada quando for feito o programa para o acionamento das bombas d'água. Uma tabela verdade pode ser montada e ela pode ser vista na Tabela 2.11. Observe que a saída foi chamada de "almi" como sendo uma variável interna que será utilizada no programa de acionamento das bombas d'água. O símbolo "1" é utilizado para condição verdadeira e o símbolo "0" é utilizado para condição falsa.

D	almc	almr	almi
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

Tabela 2.11 – Tabela verdade para "almi".

A equação booleana para esta tabela verdade é: $almi = \overline{almc}.almr + almc.\overline{almr} + almc.almr$

Esta equação pode ser simplificada por manipulação algébrica booleana ou mapa de Karnaugh e a equação simplificada é: almi = almc + almr

- A linha D0 indica a não existência de qualquer um dos alarmes.
- As outras linhas indicam a existência de, pelo menos, um dos alarmes.

O diagrama em linguagem Ladder pode ser visto na Figura 2.79.

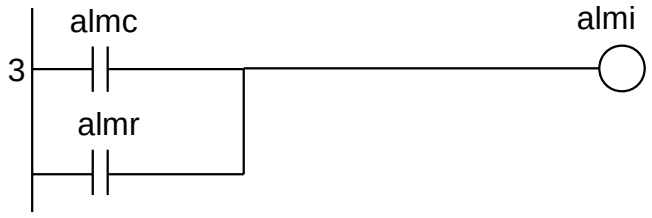

Figura 2.9 – Diagrama Ladder para a **Tabela 2.11**.

Um temporizador de saída pulsante permite que a saída esteja em modo ligado e desligado de maneira alternada quando um comando é ativado. Esse tipo de operação pode ser vista na Figura 2.10, onde o comando é ativado no instante 1 segundo e desativado no instante 5,2 segundos. A saída é ligada por 0,5 segundos e desligada por 0,5 segundos.

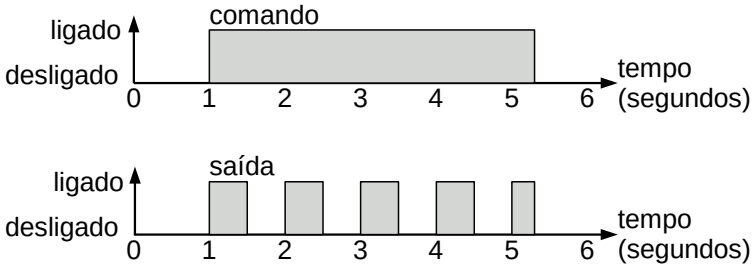

Figura 2.10 – Diagrama de tempos para um temporizador de saída pulsante.

Para fazer uma saída ser do tipo pulsante ou ligado / desligado de modo alternado os fabricantes de CLP normalmente incluem um temporizador de pulsos. Esses temporizadores podem ser configurados para que o tempo de ligado seja igual ao tempo de desligado ou permitem quem se possa configurar valores de tempo diferentes para ligado e desligado.

De maneira genérica este temporizador de pulsos pode ser visto na Figura 2.11, onde "comando" é o contato que ativa o temporizador de pulsos (TP) nomeado como "temporizador 1". A bobina de saída, nomeada como "saída", é alternada por dois valores de ligado e desligado de 1 segundo, configurados no temporizador de pulsos.

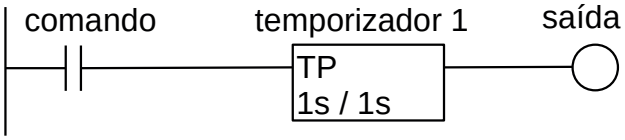

Figura 2.11 – Diagrama Ladder para o temporizador de pulsos.

Nem todo fabricante incorpora o temporizador de pulsos no programa de configuração do CLP, mas os fabricantes incorporam os temporizadores do tipo padrão: temporizador para ligar (TON) e temporizador para desligar (TOFF).

Uma solução simples utilizando apenas os temporizadores para ligar (TON) para conseguir implementar o temporizador de pulsos é mostrada na Figura 2.12. Nesta montagem quando o contato "comando" for verdadeiro, torna verdadeiro a "saída" e ativa os temporizadores para ligar (TON) "temp1" e "temp2". Assim que o temporizador "temp1" terminar a contagem (1 segundo) ele torna-se verdadeiro e o contato normalmente fechado "temp1" torna-se falso, desligando a "saída". O temporizador "temp2" continua a temporização até o término (2 segundos) então ele torna-se verdadeiro. O contato normalmente fechado "temp2" torna-se falso e desliga, momentaneamente, os dois contadores "temp1" e "temp2" e o contato normalmente fechado "temp1" torna-se verdadeiro, reiniciando o ciclo de ligado / desligado. Então o valor de "temp1" determina o tempo de "ligado" e o valor de "temp2" menos o valor de "temp1" determina o tempo de "desligado".

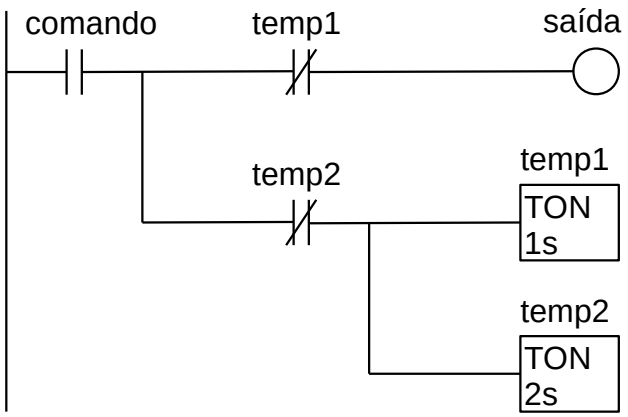

Figura 2.12 – Diagrama Ladder para saída pulsante usando temporizadores para ligar (TON).

Para o nosso problema o contato de comando é a saída “almr” que, quando verdadeiro, deve tornar pulsante uma saída nomeada como “pisca”. Essa solução pode ser vista na Figura 2.13, utilizando o temporizador de pulsos, e na Figura 2.14, utilizando o temporizador para ligar (TON). Foi adotado 1 segundo para ligado e 1 segundo para desligado.

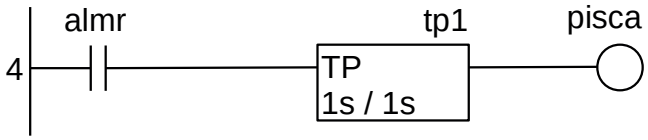

Figura 2.13 – Diagrama Ladder para “pisca” usando o temporizador de pulsos.

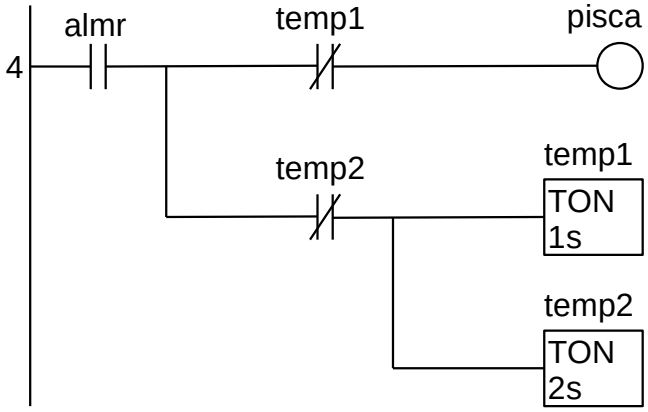

Figura 2.14 – Diagrama Ladder para “pisca” usando temporizadores para ligar (TON).

Para finalizar, o alarme “Alm” deve ser a combinação da saída interna “almi” com a saída interna “pisca”. Uma tabela verdade pode ser montada e ela pode ser vista na Tabela 2.12. O símbolo “1” é utilizado para condição verdadeira e o símbolo “0” é utilizado para condição falsa.

D	almi	pisca	Alm
0	0	0	0
1	0	1	X
2	1	0	1
3	1	1	0

Tabela 2.12 – Tabela verdade para “Alm”.

A equação booleana para esta tabela verdade é: Alm = almi.pisca

Esta equação não precisa ser simplificada por manipulação algébrica booleana ou mapa de Karnaugh.

- A linha D0 indica a não existência de alarme “Alm”.
- A linha D1 é impossível de acontecer, pois “pisca” só pode ser verdadeiro se “almi” também for verdadeiro.
- A linha D2 indica que quando “almi” for verdadeiro e “pisca” estiver no estado de desligado, então a saída de alarme “Alm” deve ser ativada.
- A linha D3 indica que quando “almi” for verdadeiro e “pisca” estiver no estado de ligado, então a saída de alarme “Alm” deve ser desativada.

O diagrama em linguagem Ladder pode ser visto na Figura 2.15.

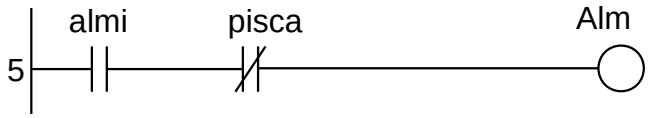

Figura 2.15 – Diagrama Ladder para a **Tabela 2.12**.

Para iniciar o programa de controle das bombas d’água, antes é necessário definir o funcionamento das memórias do nível de água da metade inferior e da metade superior da caixa d’água.

De acordo com a descrição de funcionamento 3 e 4, quando o nível de água na caixa d’água estiver entre os sensores “SNa” e “SNm” uma memória etiquetada como “alto” deve ser verdadeira. O mesmo procedimento quando o nível de água na caixa d’água estiver entre os sensores “SNm” e “SNb” uma memória etiquetada como “baixo” deve ser verdadeira.

Com o nível de água na caixa d’água aumentando, tem-se que:

- Se “SNm” passar de falso para verdadeiro, então “baixo” torna-se verdadeiro.
- Se “SNa” passar de falso para verdadeiro, então “alto” torna-se verdadeiro.

Com o nível de água na caixa d’água diminuindo, tem-se que:

- Se “SNm” passar de verdadeiro para falso, então “alto” torna-se falso.
- Se “SNb” passar de verdadeiro para falso, então “baixo” torna-se falso.

Para implementar essas definições de memória em linguagem Ladder existem duas possibilidades: o uso de bobinas de retenção (“Set” e “Reset”) ou o uso de bobina simples com contato de “selo”.

As bobinas de retenção (“Set” e “Reset”) são disponibilizadas na maioria dos controladores industriais e são representadas por uma bobina com um símbolo interno. Para a função de “Set” é comum o uso da letra “S” ou a letra “L” (para “latch”) ou uma seta apontando para cima (↑). Para a função de “Reset” é comum o uso da letra “R” ou da letra “U” (para “unlatch”) ou uma seta apontando para baixo (↓). Um exemplo de uso da bobina de retenção é mostrado no diagrama Ladder na Figura 2.16, onde quando o contato etiquetado como “ligar” for verdadeiro, torna verdadeiro a bobina etiquetada como “saída” mesmo que o contato “ligar” torne-se falso. Quando o contato etiquetado como “desligar” for verdadeiro, torna falso a bobina etiquetada como “saída” mesmo que o contato “desligar” torne-se falso.

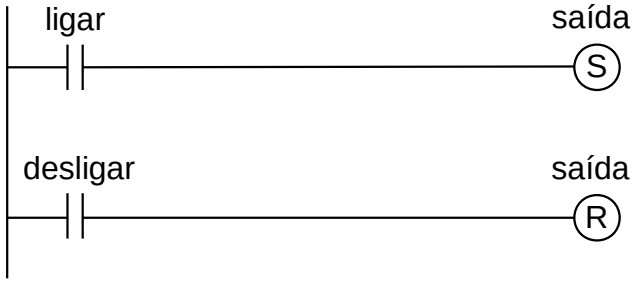

Figura 2.16 – Diagrama Ladder para memória com bobinas de Set e Reset.

A técnica de bobina simples com contato de "selo" é mostrada no diagrama Ladder na Figura 2.17, onde quando o contato etiquetado como "ligar" for verdadeiro, torna verdadeiro a bobina etiquetada como "saída" mesmo que o contato "ligar" torne-se falso, pois o contato etiquetado como "saída" será verdadeiro na próxima execução do programa. Quando o contato etiquetado como "desligar" for falso, torna falso a bobina etiquetada como "saída" mesmo que o contato "desligar" torne-se verdadeiro, pois o contato etiquetado como "saída" será falso na próxima execução do programa.

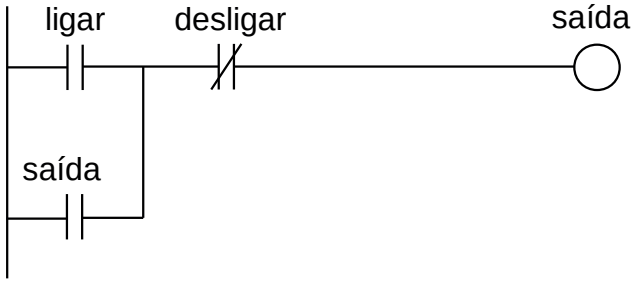

Figura 2.17 – Diagrama Ladder para memória com bobina simples e contato de selo.

Observação: para os dois casos apresentados, os contatos etiquetados como "ligar" e "desligar" não devem ser ativos simultaneamente.

Para o nosso problema, a memória "baixo" é mostrada no diagrama Ladder na Figura 2.18, usando bobinas de Set e Reset.

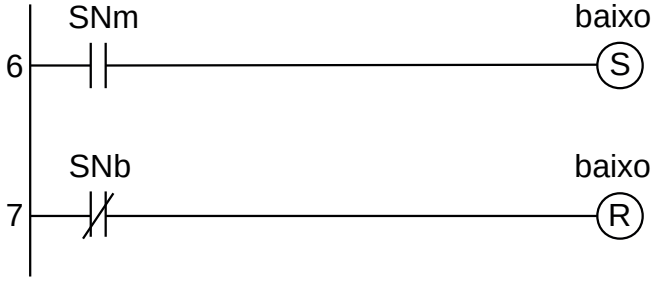

Figura 2.18 – Diagrama Ladder para memória "baixo" com bobinas de Set e Reset.

A memória "alto" é mostrada no diagrama Ladder na Figura 2.19, usando bobinas de Set e Reset.

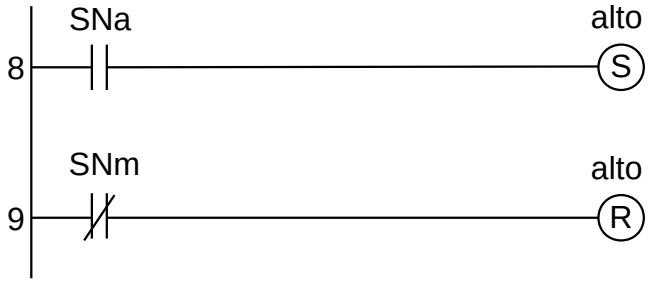

Figura 2.19 – Diagrama Ladder para memória “alto” com bobinas de Set e Reset.

Alternativamente, a memória “baixo” é mostrada no diagrama Ladder na Figura 2.20, usando bobina simples e contato de selo.

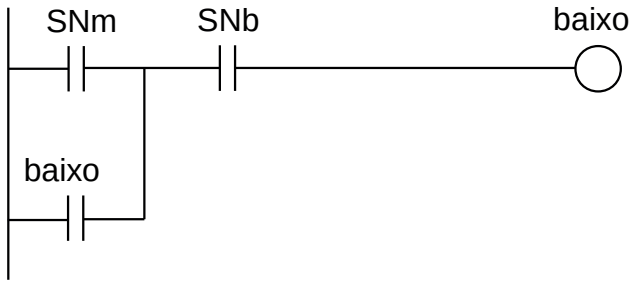

Figura 2.20 – Diagrama Ladder para memória “baixo” com bobina simples e selo.

A memória “alto” é mostrada no diagrama Ladder na Figura 2.21, usando bobina simples e contato de selo.

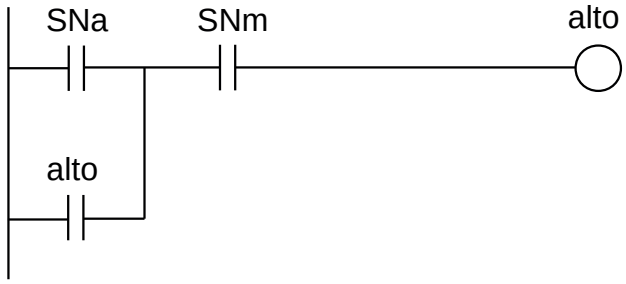

Figura 2.21 – Diagrama Ladder para memória “alto” com bobina simples e selo.

De acordo com a descrição de funcionamento 2 é necessário a criação de uma variável interna que diferencie as duas bombas d’água. É criado a variável interna com o nome de “Piloto” para o seguinte comportamento:

- Se “Piloto” é “falso” então a bomba B1 é considerada principal e a bomba B2 é considerada secundária.
- Se “Piloto” é “verdadeiro” então a bomba B2 é considerada principal e a bomba B1 é considerada secundária.

Vamos definir que o conteúdo da variável “Piloto” tenha origem em um dispositivo externo, que pode ser uma simples chave.

A tabela de entradas deve ser modificada para acrescentar esse novo dispositivo. Essa modificação pode ser vista na Tabela 2.13 com o acréscimo de uma nova linha.

Parafuso	Etiqueta	Descrição
I06	Chave	Chave seletora liga/desliga.

Tabela 2.13 – Acréscimo de linha na **Tabela 2.7**.

A entrada “Chave” quando em “falso” indica que a bomba B1 pode ser a “Piloto” se tiver água no reservatório 1, caso contrário a bomba B2 será a “Piloto”. A entrada “Chave” quando em “verdadeiro” indica que a bomba B2 pode ser a “Piloto” se tiver água no reservatório 2, caso contrário a bomba B1 será a “Piloto”. A variável “Piloto” é definida pela combinação das variáveis “Chave”, “R1b” e “R2b”. Uma tabela verdade pode ser montada e ela pode ser vista na Tabela 2.14. O símbolo “1” é utilizado para condição verdadeira e o símbolo “0” é utilizado para condição falsa.

D	Chave	R1b	R2b	Piloto	Bomba
0	0	0	0	X	X
1	0	0	1	1	B2
2	0	1	0	0	B1
3	0	1	1	0	B1
4	1	0	0	X	X
5	1	0	1	1	B2
6	1	1	0	0	B1
7	1	1	1	1	B2

Tabela 2.14 – Tabela verdade para “piloto”.

A equação booleana para esta tabela verdade é:

$$\text{Piloto} = \overline{\text{Chave}}.\overline{\text{R1b}}.\text{R2b} + \text{Chave}.\overline{\text{R1b}}.\text{R2b} + \text{Chave.R1b.R2b}$$

Esta equação pode ser simplificada utilizando manipulação algébrica booleana ou mapa de Karnaugh.

- As linhas D0 e D4 não importam pois se faltar água nos dois reservatórios será dado alarme e nenhuma das bombas deve funcionar.
- A linha D1 indica que a chave seleciona a bomba B1, mas como falta água no reservatório 1 e tem água no reservatório 2, então a bomba piloto é a B2.
- As linhas D2 e D3 indicam que a chave seleciona a bomba B1 e tem água no reservatório 1.
- As linhas D5 e D7 indicam que a chave seleciona a bomba B2 a tem água no reservatório 2.
- A linha D6 indica que a chave seleciona a bomba B2, mas como falta água no reservatório 2 e tem água no reservatório 1, então a bomba piloto é a B1.

O diagrama em linguagem Ladder, não simplificado, pode ser visto na Figura 2.22.

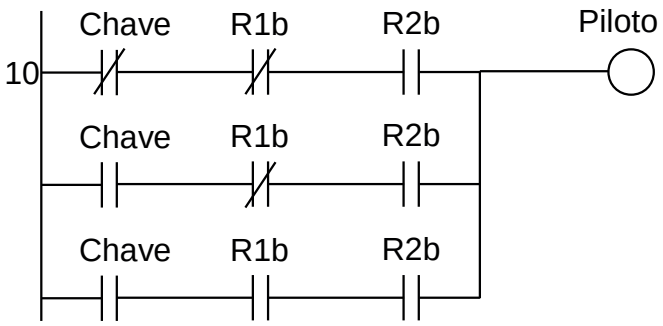

Figura 2.22 – Diagrama Ladder para a **Tabela 2.14**.

Para implementar o controle das bombas d'água é necessário que não tenha alarme ("almi" falso) e que tenha água no reservatório correspondente e seja dependente da combinação das variáveis de nível na caixa d'água "baixo" e "alto" e da variável de seleção de bomba "Piloto". Uma tabela verdade pode ser montada e ela pode ser vista na Tabela 2.15 para a bomba B1 e na Tabela 2.16 para a bomba B2. O símbolo "1" é utilizado para condição verdadeira e o símbolo "0" é utilizado para condição falsa.

Linha	almi	R1b	alto	baixo	Piloto	B1
1	0	1	0	0	0	1
2	0	1	0	0	1	1
3	0	1	0	1	0	1
4	0	1	0	1	1	0
5	0	1	1	0	0	X
6	0	1	1	0	1	X
7	0	1	1	1	0	0
8	0	1	1	1	1	0
9	1	X	X	X	X	0
10	X	0	X	X	X	0

Tabela 2.15 – Tabela verdade para a bomba "B1".

A equação booleana para esta tabela verdade é:

$$B1 = \overline{almi}.R1b.\overline{alto}.\overline{baixo}.\overline{Piloto} + \overline{almi}.R1b.\overline{alto}.\overline{baixo}.Piloto + \overline{almi}.R1b.\overline{alto}.baixo.\overline{Piloto}$$

Esta equação pode ser simplificada utilizando manipulação algébrica booleana ou mapa de Karnaugh.

Uma simplificação algébrica pode ser:

$$B1h = \overline{almi}.R1b.\overline{alto}$$

$$B1 = B1h.(\overline{baixo}.\overline{Piloto} + \overline{baixo}.Piloto + baixo.\overline{Piloto})$$

- As linhas 1 e 2 indicam que a caixa d'água está vazia, e, independente de "Piloto", a bomba d'água B1 deve ser acionada.
- A linha 3 indica que a caixa d'água tem água da metade para cima e "Piloto" seleciona a bomba B1.
- A linha 4 indica que a caixa d'água tem água da metade para cima e "Piloto" seleciona a bomba B2.

- As linhas 5 e 6 são impossíveis de acontecer.
- As linhas 7 e 8 indicam que a caixa d'água está totalmente cheia e a bomba B1 não é ligada.
- A linha 9 indica que se existir alarme e independe das outras variáveis, as bombas não são ligadas.
- A linha 10 indica que se não tiver água no reservatório 1 a bomba B1 não deve ser ligada.

O diagrama em linguagem Ladder, com a simplificação proposta, pode ser visto na Figura 2.23.

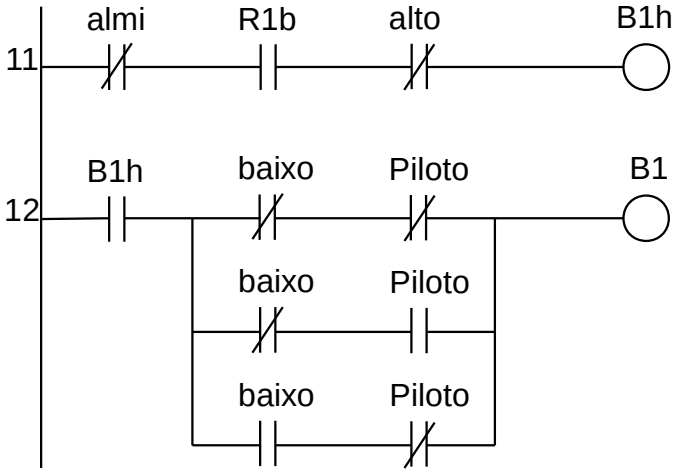

Figura 2.23 – Diagrama Ladder para o controle da bomba B1.

Linha	almi	R2b	alto	baixo	Piloto	B2
1	0	1	0	0	0	1
2	0	1	0	0	1	1
3	0	1	0	1	0	0
4	0	1	0	1	1	1
5	0	1	1	0	0	X
6	0	1	1	0	1	X
7	0	1	1	1	0	0
8	0	1	1	1	1	0
9	1	X	X	X	X	0
10	X	0	X	X	X	0

Tabela 2.16 – Tabela verdade para a bomba "B2".

A equação booleana para esta tabela verdade é:

$$B2 = \overline{almi}.R2b.\overline{alto}.\overline{baixo}.\overline{Piloto} + \overline{almi}.R2b.\overline{alto}.\overline{baixo}.Piloto + \overline{almi}.R2b.\overline{alto}.baixo.Piloto$$

Esta equação pode ser simplificada utilizando manipulação algébrica booleana ou mapa de Karnaugh.

Uma simplificação algébrica pode ser:

$$B2h = \overline{almi}.R2b.\overline{alto}$$

$$B2 = b2h.(\overline{baixo}.\overline{Piloto} + \overline{baixo}.Piloto + baixo.Piloto)$$

- As linhas 1 e 2 indicam que a caixa d'água está vazia, e, independente de "Piloto", a bomba d'água B2 deve ser acionada.
- A linha 3 indica que a caixa d'água tem água da metade para cima e "Piloto" seleciona a bomba B1.
- A linha 4 indica que a caixa d'água tem água da metade para cima e "Piloto" seleciona a bomba B2.
- As linhas 5 e 6 são impossíveis de acontecer.
- As linhas 7 e 8 indicam que a caixa d'água está totalmente cheia e a bomba B2 não é ligada.
- A linha 9 indica que se existir alarme e independe das outras variáveis, as bombas não são ligadas.
- A linha 10 indica que se não tiver água no reservatório 2 a bomba B2 não deve ser ligada.

O diagrama em linguagem Ladder, com a simplificação proposta, pode ser visto na Figura 2.24.

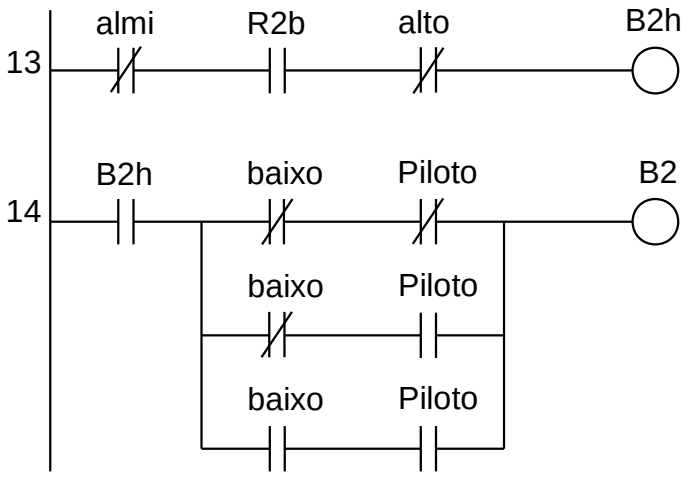

Figura 2.24 – Diagrama Ladder para o controle da bomba B2.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às Tabelas 2.7 e 2.8. A tabela com as variáveis internas pode ser vista na Tabela 2.18.

Variável	Etiqueta	Descrição
M1	almc	Alarme dos sensores da caixa d'água.
M2	almr	Alarme dos sensores dos reservatórios.
M3	almi	Alarme geral.
M4	pisca	Pulso alternado ligado / desligado.
M5	baixo	Nível de água do meio para baixo.
M6	alto	Nível de água do meio para o alto.
M7	Piloto	Identificador de bomba d'água principal.
M8	B1h	Habilitação da bomba d'água 1.
M9	B2h	Habilitação da bomba d'água 2.
TP1	tp1	Temporizado de pulsos 1s/1s.

Tabela 2.18 – Descrição das variáveis internas.

2.3 – Solução de problemas utilizando frases lógicas

As frases lógicas podem ser escritas da seguinte maneira:

Se (lógica de entradas for verdadeira) **então** (dispositivo de saída é verdadeiro ou falso).

Para a parte de "lógica das entradas" considere as entradas digitais dos sensores, valores digitais de registradores internos como temporizadores e contadores e valores de registradores para comparação de tamanho. A negação de um valor de entrada também deve ser considerado.

Para a parte de "dispositivo de saída" considere as saídas digitais dos atuadores, valores digitais de registradores internos como temporizadores e contadores e armazenamento de valores em registradores de tamanho. A negação de um valor de saída também pode ser usado.

2.3.1 – Exemplo 4. Tanque de mistura de líquidos

Este exemplo é um processo comum e simples. É feita uma análise em relação aos sensores para determinar as condições de ativação e desativação de cada saída.

Considere o problema de automatizar um tanque de mistura de dois líquidos conforme mostrado na Figura 2.25.

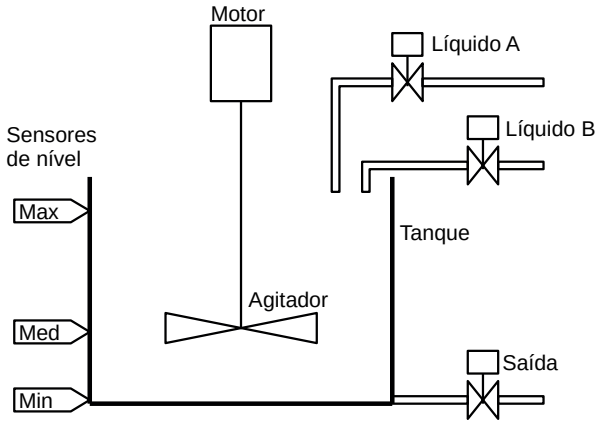

Figura 2.25 – Tanque de mistura de dois líquidos.

Descrição da máquina ou processo

Essa máquina possui um tanque com um agitador, para mistura de líquidos, acionado por um motor. Este motor é ativado através de um contator. Existem 3 válvulas solenoide para controle de entrada e saída do líquido e 3 sensores de nível. Os sensores são verdadeiros se o líquido estiver presente no sensor.

Descrição operacional da máquina ou processo

A operação da máquina é: primeiro coloca-se o Líquido A até o sensor Med ser verdadeiro, depois coloca-se o Líquido B até o sensor Max ser verdadeiro, depois liga-se o Motor por um tempo de 30 segundos, depois abre-se a válvula de Saída até o sensor Min ser falso.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na Tabela 2.19.

Parafuso	Etiqueta	Descrição
I01	Min	Sensor de nível do tipo boia.
I02	Med	Sensor de nível do tipo boia.
I03	Max	Sensor de nível do tipo boia.

Tabela 2.19 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na Tabela 2.20.

Parafuso	Etiqueta	Descrição
Q01	Motor	Contator de acionamento do motor.
Q02	Líquido A	Válvula solenoide.
Q03	Líquido B	Válvula solenoide.
Q04	Saída	Válvula solenoide.

Tabela 2.20 – Descrição das saídas.

Solução por frases lógicas

Controle da válvula solenoide para o "Líquido A":

1 – Se o sensor "Min" for falso então fazer a variável "Encher" verdadeiro e memorizar e o solenoide "Líquido A" verdadeiro e memorizar.

Início do processo. O tanque está vazio. O solenoide "Líquido A" abre e o líquido começa a encher o tanque. Com o nível do líquido subindo acima do sensor "Min" este torna-se verdadeiro. A variável "Encher" é utilizado para colocar o líquido B e evitar a abertura do solenoide "Líquido B" quando o tanque estiver esvaziando. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.26.

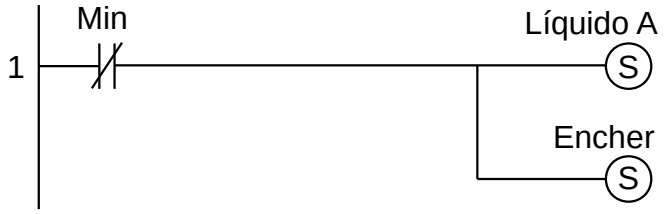

Figura 2.26 – Diagrama Ladder para a frase lógica 1.

2 – Se o sensor "Med" for verdadeiro então fazer o solenoide "Líquido A" falso.

Quando o líquido chegar no sensor “Med” indica que a quantidade certa do líquido A já foi colocada no tanque de mistura e o solenoide “Líquido A” deve ser fechado. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.27.

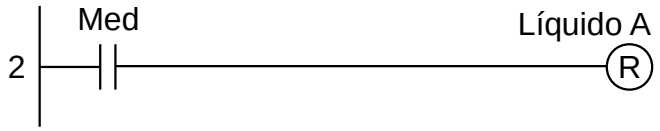

Figura 2.27 – Diagrama Ladder para a frase lógica 2.

Controle da válvula solenoide para o “Líquido B”:

> 3 – Se o sensor “Med” for verdadeiro e a variável “Encher” for verdadeiro e o solenoide “Líquido B” for falso então fazer o solenoide “Líquido B” verdadeiro e memorizar.

Para a colocação do líquido B é necessário o uso de uma variável que passe da condição falso para a condição verdadeiro para ativação da memória “Líquido B”. Quando a variável “Encher” for falso indica que o tanque está esvaziando. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.28.

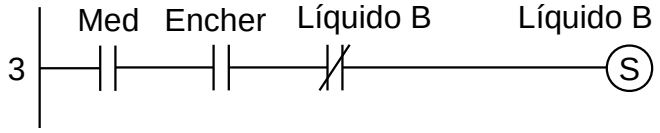

Figura 2.28 – Diagrama Ladder para a frase lógica 3.

> 4 – Se o sensor “Max” for verdadeiro então fazer o solenoide “Líquido B” falso e a variável “Encher” falso.

Quando o líquido chegar no sensor “Max” indica que a quantidade certa do líquido B já foi colocada no tanque de mistura e o solenoide “Líquido B” deve ser fechado. O tanque está cheio. A variável “Encher” torna-se falso para prevenir que não se coloque o líquido B enquanto o tanque estiver esvaziando. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.29.

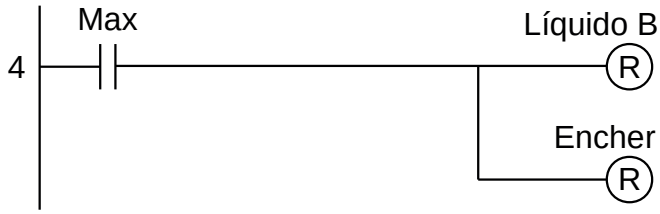

Figura 2.29 – Diagrama Ladder para a frase lógica 4.

Controle do “Motor” do agitador:

> 5 – Se o sensor “Max” for verdadeiro então fazer o “Motor” verdadeiro e o “Temporizador” de 30 segundos verdadeiro.

Quando a quantidade certa dos dois líquidos for colocada no tanque de mistura, deve-se ativar o motor do agitador e o temporizador de 30 segundos. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.30.

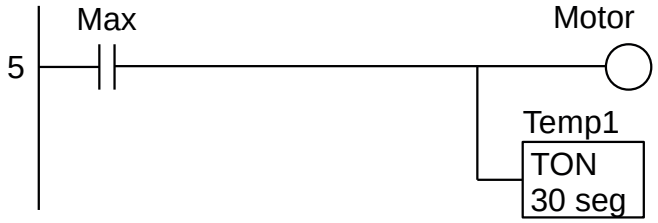

Figura 2.30 – Diagrama Ladder para a frase lógica 5.

Controle da válvula solenoide de "Saída":

6 – Se a saída do "Temporizador" for verdadeiro então fazer o solenoide "Saída" verdadeiro e memorizar.

Quando o temporizador terminar a contagem de 30 segundos a saída do temporizador é verdadeira, a válvula solenoide "Saída" é ativada e o tanque começa a esvaziar. Então o sensor "Max" torna-se falso e desliga o motor e o temporizador. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.31.

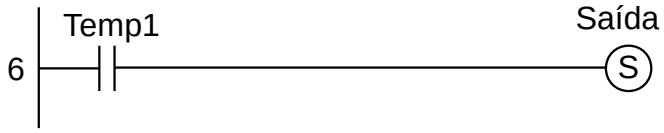

Figura 2.31 – Diagrama Ladder para a frase lógica 6.

7 – Se o sensor "Min" for falso então fazer o solenoide "Saída" falso.

Fim do processo. O tanque está vazio. Então deve-se desligar o solenoide "Saída". O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.32.

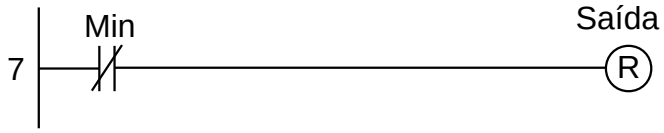

Figura 2.32 – Diagrama Ladder para a frase lógica 7.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 2.19 e 2.20. A tabela com as variáveis internas pode ser vista na Tabela 2.21.

Variável	Etiqueta	Descrição
M1	Encher	Memória para tanque enchendo.
T1	temp1	Temporizador para ligar 30 segundos.

Tabela 2.21 – Descrição das variáveis internas.

2.3.2 – Exemplo 5. Máquina seladora térmica

Outro exemplo de uma máquina um pouco mais complexa: uma máquina seladora térmica automática. O diagrama esquemático desta máquina pode ser visto na Figura 2.33.

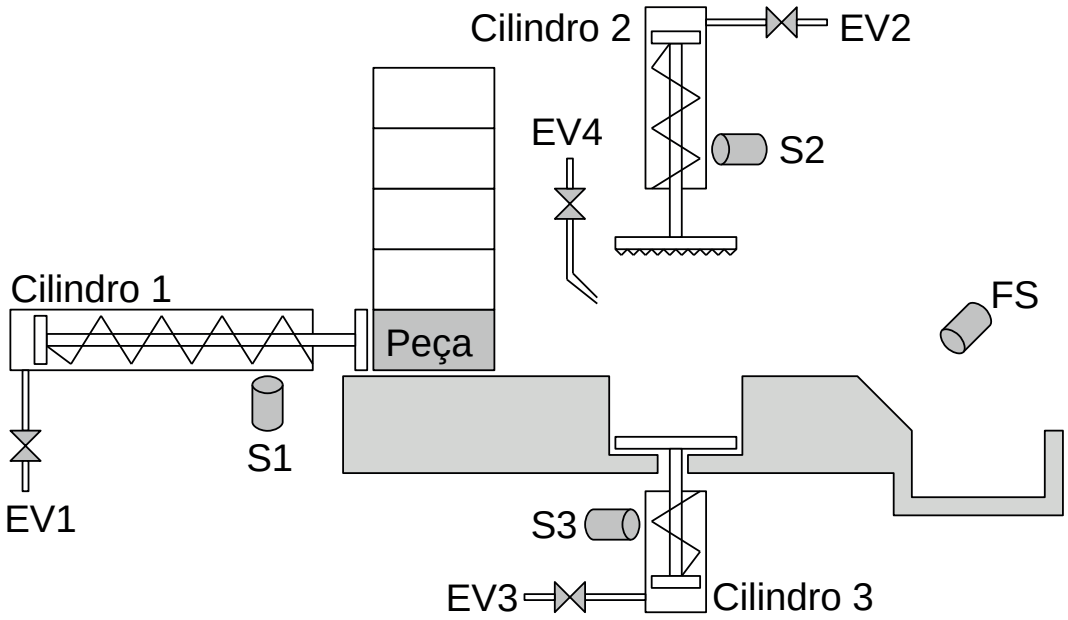

Figura 2.33 – Máquina seladora térmica de peças.

Descrição da máquina ou processo

Essa máquina tem 4 eletroválvulas EV1, EV2, EV3 e EV4 e 4 sensores S1, S2, S3 e FS. As eletroválvulas EV1, EV2 e EV3 são do tipo 3/2 vias para acionamento de cilindros pneumáticos de simples ação com retorno por mola. A eletroválvula EV4 permite que o ar comprimido seja projetado como um jato de ar de alta pressão para empurrar uma peça. Os sensores S1, S2 e S3 são chave tipo "reed" que são acionadas pelo magnetismo presente no embolo dos cilindros pneumáticos. O sensor etiquetado como FS é do tipo ótico por reflexão passiva e emite um sinal quando uma peça ficar na frente dele. Existe uma chave do tipo liga/desliga de giro, com etiqueta "Ligar" para energização do controlador digital. Existe uma botoeira, com etiqueta "Partida", para iniciar o processo automático.

Descrição operacional da máquina ou processo

Quando a chave "Ligar" estiver ligada e a botoeira "Partida" ser pressionada, a máquina deve funcionar de modo ininterrupto. Quando a chave "Ligar" for desligada o processo deve parar no início. Deve ser garantido que a botoeira "Partida" não seja ativa durante o funcionamento automático mas apenas para iniciar o processo.

Quando a eletroválvula EV1 abrir, o ar comprimido atua no cilindro pneumático 1 e permite empurrar a peça a ser selada desde o alimentador de peças até a região de selagem térmica, acima do cilindro 3. O fim de movimento do cilindro 1 é detectado pelo sensor S1, que faz com que a eletroválvula EV1 seja desligada e o cilindro 1 possa recuar pela ação da mola interna.

Após a peça cair na região de selagem térmica, a eletroválvula EV2 é aberta e o ar comprimido atua no cilindro pneumático 2 que permite descer o selador térmico até entrar em contato com a peça. O fim de movimento do cilindro 2 é detectado pelo sensor S2. O selador térmico na extremidade inferior do cilindro 2 deve permanecer em contato com a peça por um tempo de 20 segundos, ou seja, o cilindro

2 deve ficar acionado durante este tempo. Depois deste tempo a eletroválvula EV2 pode ser desligada e permite que o cilindro 2 possa recuar pela ação da mola interna.

O avanço do cilindro 3 pode ser feito de duas maneiras: após o cilindro 2 recuar totalmente ou simultaneamente com o cilindro 2 se o cilindro 2 recuar mais rápido que o cilindro 3. Como não existe um sensor de início de curso no cilindro 2, adotaremos a segunda maneira, ou seja, quando o sensor S2 deixar de ser verdadeiro daremos início ao movimento do cilindro 3. Então a eletroválvula EV3 é aberta e o ar comprimido atua no cilindro pneumático 3 e permite empurrar a peça já selada para cima até o mesmo nível da mesa. O fim de movimento do cilindro 3 é detectado pelo sensor S3. O cilindro 3 deve permanecer acionado até a peça ser expulsa e cair no coletor de peças à direita.

A eletroválvula EV4 é aberta quando o cilindro 3 estiver totalmente no fim do seu curso. O jato de ar então vai empurrar a peça até o coletor de peças à direita. Quando a peça passar pelo sensor FS as eletroválvulas EV3 e EV4 são desligadas, permitindo o recuo do cilindro 3 e o desligamento do jato de ar. Então o processo pode ser reiniciado.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na Tabela 2.22.

Parafuso	Etiqueta	Descrição
I01	S1	Sensor fim-de-curso magnético.
I02	S2	Sensor fim-de-curso magnético.
I03	S3	Sensor fim-de-curso magnético.
I03	FS	Sensor óptico reflexivo.
I04	Partida	Botoeira NA.
I05	Ligar	Chave liga/desliga de giro.

Tabela 2.22 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na Tabela 2.23.

Parafuso	Etiqueta	Descrição
Q01	EV1	Eletroválvula pneumática 3/2 vias.
Q02	EV2	Eletroválvula pneumática 3/2 vias.
Q03	EV3	Eletroválvula pneumática 3/2 vias.
Q04	EV4	Eletroválvula pneumática 3/2 vias.

Tabela 2.23 – Descrição das saídas.

Solução por frases lógicas

Para este problema será considerado que o programa do controlador possa ser iniciado através de uma botoeira etiquetada como “Partida”. Esta botoeira será pressionada apenas quando a máquina

estiver ligada mas parada. Após esta botoeira ser pressionada pela primeira vez, ela fica desativada após o próximo ciclo do programa do controlador. Essa técnica é conhecida como “um disparo” ou, em inglês, “one shot”. O diagrama em linguagem Ladder para esta técnica de programação pode ser visto na Figura 2.34.

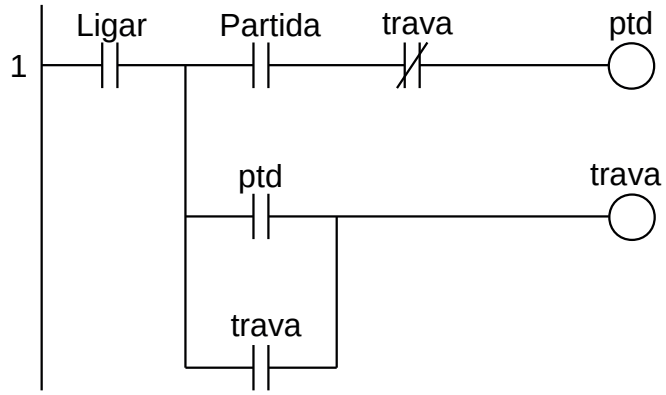

Figura 2.34 – Diagrama Ladder para técnica de “um disparo”.

Ao iniciar o controlador, as variáveis “ptd” e “trava” são falsos.

Na linha 1, quando “Ligar” for verdadeiro e “Partida” for verdadeiro e trava for falso então “ptd” é verdadeiro. Como “ptd” é verdadeiro então a variável “trava” também é verdadeiro e é feito o “selo” sobre “ptd”.

Durante todo o ciclo do programa do controlador a variável “ptd” vai ficar verdadeiro até a execução do próximo ciclo do programa do controlador, quando a variável “trava” é interpretada com verdadeiro e, de acordo com a linha 1, a variável “ptd” torna-se falsa. A variável “trava” mantém-se verdadeiro através do selo sobre o contato “ptd” até que a entrada “Ligar” torna-se falso.

Início de ciclo

A variável “ptd” será utilizada para iniciar o ciclo de automação, mas é necessário que o ciclo se repita. Conforme a descrição funcional, o ciclo termina quando a peça cai no coletor de peças e o sensor “FS” é ativado momentaneamente quando a peça passa em frente ao sensor.

Então uma frase lógica pode ser montada para definir o início de ciclo. Uma nova variável é definida: “iciclo” que é o início do ciclo.

1 – Se “ptd” for verdadeiro ou “FS” for verdadeiro e “Ligar” for verdadeiro então fazer a variável interna “iciclo” verdadeiro.

O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.35.

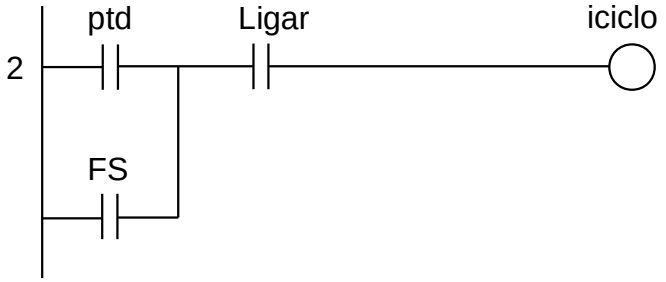

Figura 2.35 – Diagrama Ladder para frase lógica 1.

Acionamento do cilindro 1

2 – Se “iciclo” for verdadeiro então fazer a saída “EV1” verdadeiro e memorizar.

O início do ciclo de automação desta máquina começa pelo avanço do cilindro 1. Então a variável “iciclo” ativa a eletroválvula 1 “EV1” e mantém ativada pois a variável “iciclo” é do tipo momentâneo. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.36.

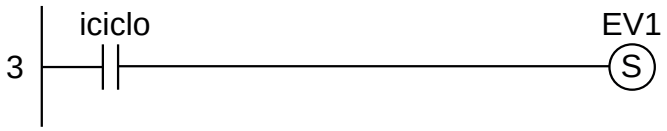

Figura 2.36 – Diagrama Ladder para frase lógica 2.

3 – Se “S1” for verdadeiro então fazer a saída “EV1” falso.

Quando a haste do cilindro 1 chegar no final do seu curso o sensor “S1” é ativado pelo magnetismo do embolo e então deve ser feito o desligamento da eletroválvula “EV1” e a haste do cilindro recua pela ação da mola. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.37.

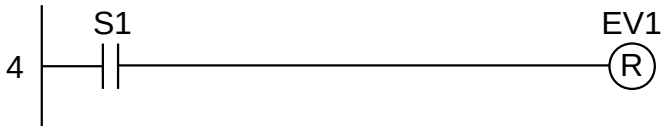

Figura 2.37 – Diagrama Ladder para frase lógica 3.

Acionamento do cilindro 2

4 – Se “S1” for verdadeiro então fazer a saída “EV2” verdadeiro e memorizar.

O final de movimento da haste do cilindro 1, indicado pelo sensor “S1”, implica que a peça está na região de selagem térmica, então o cilindro 2 pode ser ativado e descer o selador térmico que está fixado no final da haste do cilindro 2. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.38.

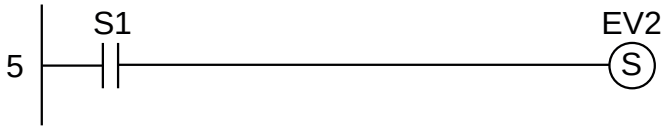

Figura 2.38 – Diagrama Ladder para frase lógica 4.

5 – Se “S2” for verdadeiro então fazer o temporizador “Temp1” de 20 segundos verdadeiro.

O final de movimento da haste do cilindro 2, indicado pelo sensor “S2”, implica que o selador térmico está em contato com a peça. De acordo com as especificações, é necessário um tempo de 20 segundos para que a selagem fique completa. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.39.

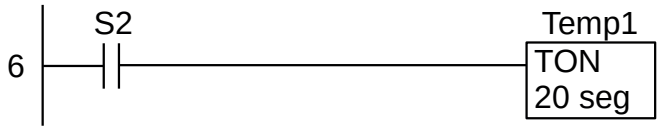

Figura 2.39 – Diagrama Ladder para frase lógica 5.

6 – Se “Temp1” for verdadeiro então fazer a saída “EV2” falso.

Quando o temporizador terminar a contagem de 20 segundos, a eletroválvula “EV2” deve ser desligada para permitir o retorno da haste do cilindro 2 pela ação da mola interna. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.40.

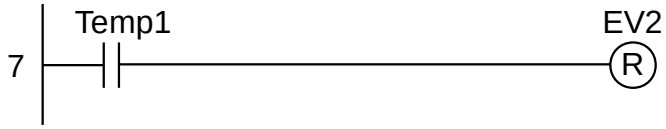

Figura 2.40 – Diagrama Ladder para frase lógica 6.

Acionamento do cilindro 3

7 – Se “Temp1” for verdadeiro então fazer a saída “EV3” verdadeiro e memorizar.

O término da contagem de tempo do temporizador também ativa a eletroválvula “EV3” para permitir que a haste do cilindro 3 retire a peça da região de selagem térmica e posicione a peça em frente ao sopro de ar comprimido. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.41.

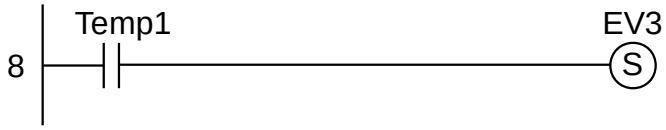

Figura 2.41 – Diagrama Ladder para frase lógica 7.

Acionamento do sopro de ar comprimido

8 – Se “S3” for verdadeiro então fazer a saída “EV4” verdadeiro.

O final de movimento da haste do cilindro 3, indicado pelo sensor “S3”, implica que a peça já saiu totalmente da região de selagem térmica e se encontra na frente do sopro de ar comprimido. A eletroválvula “EV4” é aberta e um jato de ar comprimido vai empurrar a peça para o coletor de peças à direita. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.42.

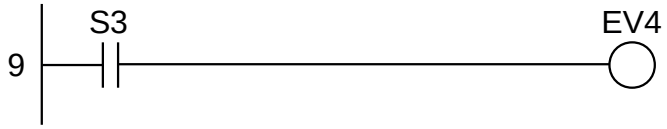

Figura 2.42 – Diagrama Ladder para frase lógica 8.

9 – Se “FS” for verdadeiro então fazer a saída “EV3” falso.

Quando a peça passar em frente ao sensor óptico “FS” implica que a peça está caindo em direção ao coletor de peças à direita e a eletroválvula “EV3” deve ser desligada, permitido o recuo da haste do cilindro 3 pela ação da mola interna, e a eletroválvula “EV4” é desligada quando o cilindro 3 começar a recuar e o sensor “S3” for falso, cortando o jato de ar comprimido. O diagrama em linguagem Ladder para esta frase lógica pode ser visto na Figura 2.43.

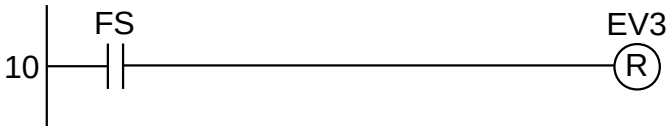

Figura 2.43 – Diagrama Ladder para frase lógica 9.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 2.22 e 2.23. A tabela com as variáveis internas pode ser vista na Tabela 2.24.

Variável	Etiqueta	Descrição
M1	ptd	Partida do programa tipo “um disparo”.
M2	trava	Memória de trava para “Partida”.
M3	iciclo	Memória de início de ciclo de automação.
T1	temp1	Temporizador para ligar 20 segundos.

Tabela 2.24 – Descrição das variáveis internas.

2.4 – Conclusão

O método por álgebra booleana é aplicável para todo tipo de máquina ou processo. Existem dois tipos de processos: combinacionais e sequenciais. Os processos combinacionais associam as entradas diretamente com as saídas, ou seja, uma mudança numa entrada implica numa mudança imediata nas saídas correspondentes. Eventualmente uma variável interna é necessária ser criada para facilitar a montagem das tabelas verdade ou para definir uma condição de funcionamento. Esse problema pode ser de difícil visualização por parte do programador. Os processos sequenciais, tanto da máquina de Mealy quanto da máquina de Moore, necessitam de uma etapa de memória que vai reter o “estado” da máquina. As implementações dos processos sequenciais via tabela verdade são muito complexas e passíveis de erros.

Este método, então, para ser aplicado, passa pela seguinte análise da máquina ou procedimento:

1) O processo é linear e simples.
2) Está baseado apenas em situações lógicas.

3) As saídas estão intrinsecamente ligadas às entradas.
4) A lógica de controle pode ser representada por equações de Boole.
5) As equações podem ser simplificadas.
6) Diferentes controladores podem atuar nas saídas.

O procedimento geral para a solução de um problema é:

1) Descrição da máquina ou processo.
2) Descrição do funcionamento.
3) Definição das entradas e saídas.
4) Montagem das equações booleanas.
5) Simplificação das equações booleanas.
6) Desenho do diagrama "Ladder".
7) Simulação em computador ou com controlador real.
8) Depuração de erros.
9) Compilação e transferência.

Durante todo o procedimento para solucionar o problema é necessário a completa documentação do projeto.

2.4.1 – Exemplo 6. O problema do botão e da lâmpada

Um exemplo de um problema aparentemente simples mas que é de difícil compreensão para o programador iniciante: ligar e desligar uma lâmpada com um botão tipo "push-button".

O funcionamento é o seguinte:

1) O operador humano aperta o botão e a lâmpada acende imediatamente.
2) O operador humano continua com o dedo no botão e a lâmpada continua acesa.
3) O operador humano retira o dedo do botão e a lâmpada continua acesa.
4) O operador humano aperta, novamente, o botão e a lâmpada apaga imediatamente.
5) O operador humano continua com o dedo no botão e a lâmpada continua apagada.
6) O operador humano retira o dedo do botão e a lâmpada continua apagada.

As limitações para a solução deste problema são:

- O programa deve utilizar os comandos básicos da linguagem "Ladder": contato normal, contato inversor e bobina normal ou com memória.
- A quantidade de linhas de programação deve ser a menor possível.

Uma das soluções para esse problema pode ser visto na Figura 2.44. A variável interna "aceso" é verdadeiro quando a lâmpada estiver acesa e o dedo estiver pressionando o botão. A variável interna "apagado" é verdadeiro quando a lâmpada estiver apagada e o dedo estiver pressionando o botão. Ao retirar o dedo do botão as variáveis internas são falsas. O leitor pode tentar montar as tabelas verdades para as três saídas e comparar o funcionamento deste programa com os programas para flip-flop tipo T. Esse problema será visto de maneira mais fácil em outros métodos de solução de problemas.

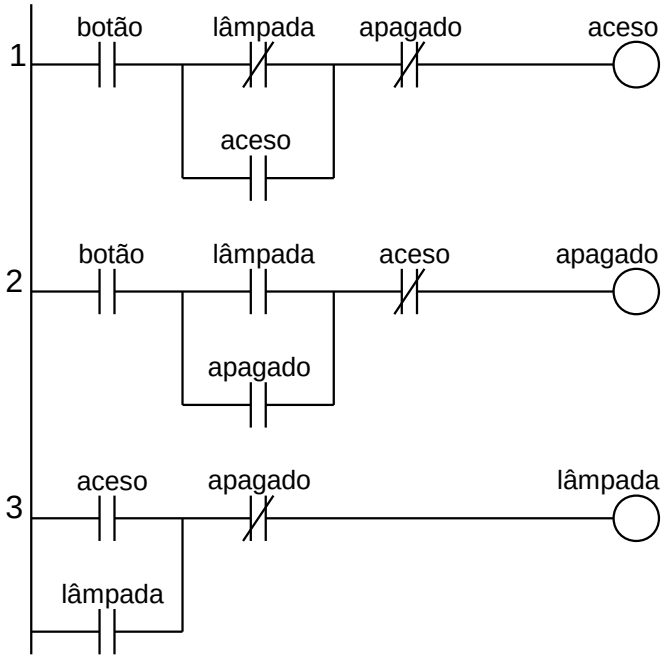

Figura 2.44 – Diagrama Ladder para o problema do botão e lâmpada (versão 1).

Na Figura 2.45 pode ser visto outra solução para o problema do botão e da lâmpada considerando o uso da técnica de "um pulso" da botoeira, ou seja, uma saída interna associada à botoeira é verdadeira apenas no primeiro ciclo do programa que identifica que a botoeira foi pressionada.

Na linha 1, a bobina "mem1" é verdadeiro se o botão for pressionado (verdadeiro) e "mem2" é verdadeiro, porque a linha 2 ainda não foi executada. Na segunda execução do programa a bobina "mem1" é falso porque o contato "mem2" é falso.

Na linha 2, a bobina "mem2" é verdadeira se o botão for pressionado (verdadeiro).

Na linha 3, a bobina "lâmpada" será verdadeira se "mem1" for verdadeiro e "lâmpada" (lâmpada apagada) for verdadeiro. A bobina "lâmpada" será falso se "mem1" for verdadeiro e "lâmpada" (lâmpada acesa) for verdadeiro.

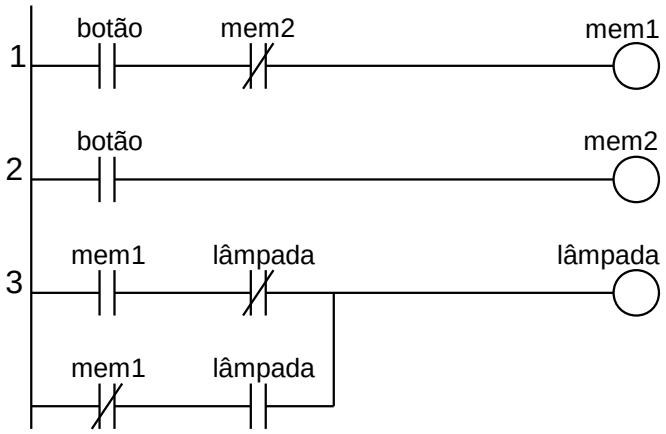

Figura 2.45 – Diagrama Ladder para o problema do botão e lâmpada (versão 2).

Capítulo 3
Estruturas de modelagem de eventos discretos

É apresentado uma visão geral de várias abordagens para modelagem de eventos discretos e suas variações. É apresentado tanto as estruturas textuais quanto as estruturas gráficas.

3.1 – Introdução

Para uma programação efetiva faz-se necessária a criação de um modelo representativo do funcionamento do processo ou máquina. O detalhamento dos elementos e de como estes elementos interagem com a evolução dos eventos é fundamental para a compreensão humana e para facilitar a programação em uma determinada linguagem computacional. Os eventos discretos podem evoluir através de intervalos de tempo, ou através de elementos de detecção de um parâmetro, ou através de condições internas.

Os estudos em álgebra booleana sequencial permite, através da Máquina de Estados Finitos, o gerenciamento das saídas em relação às entradas utilizando dois elementos fundamentais: uma memória de estado e um pulso de troca de estado (em inglês: "clock"). Existe também uma lógica combinacional que define qual o próximo estado através da observação das entradas e uma lógica combinacional que define como as saídas são representadas de acordo com os estados. As duas representações conhecidas, Máquina de Moore e Máquina de Mealy, são usadas para a geração de saídas baseadas em sequências de entradas, entretanto, apesar da máquina de Mealy ser mais fácil de projetar e gastar menos dispositivos, a máquina de Moore apresenta uma melhor estabilidade quanto ao projeto de programas para controladores de processos discretos.

Os eventos discretos normalmente encontrados em aplicações industriais podem ser do tipo:

a) Sequência simples.
b) Seleção de sequências simples.
c) Sequências com acumulação de valores.
d) Sequências com alteração de evolução de eventos.
e) Sequências complexas com eventos por gatilho.

Uma sequência de eventos discreto é considerada "simples" quando os eventos estão numa ordem que não muda. Essa ordem pode ser representada conforme mostrado na figura 3.1. Os eventos possuem uma condição de estar da máquina ou processo e uma condição de transição de um evento para outro. Normalmente a condição de estar significa um evento de ação ou movimento da máquina, e a transição significa um evento de lógica que permite a máquina passar de uma condição para outra condição.

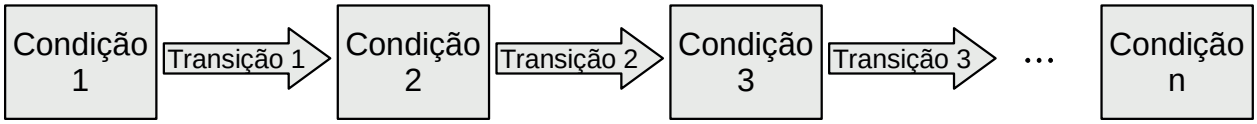

Figura 3.1 – Representação gráfica de uma sequência de eventos.

Normalmente os sistemas industriais são repetitivos e então existe uma transição do último evento para o primeiro evento. A transição de um evento para outro está associado a uma condição lógica relativo a entradas ou parâmetros internos.

Um sistema é dito de seleção de sequências quando, a máquina ou processo estiver em uma determinada condição, ocorrer um evento de seleção de sequências e da condição inicial existir mais de um evento de transição. Esses eventos de transição múltipla podem ser por escolha (lógica booleana OU) ou cumulativo (lógica booleana E).

As sequências com acumulação de valores tendem a alterar parâmetros de um evento tanto na funcionalidade deste evento quanto na lógica da transição. Essa alteração de parâmetros tende a modificar as diversas transições, alterando o caminho de execução das tarefas da máquina.

As sequências mais complexas permitem uma tomada de decisão com possibilidade de aprendizado e alteração das condições das transições. Sistemas especialistas podem combinar as ocorrências de eventos de transição e gerar novas conexões entre as condições operacionais da máquina. Isso é muito útil na realização de diagnósticos operacionais e execução de manutenção automática.ç

3.2 – Modelagem

Máquinas, processos, sistemas, ou qualquer coisa pode ser representado em um meio físico igual ou diferente do meio existente. A essa representação é dado o nome de **modelo**. Somente as partes relevantes ou de interesse da coisa física é considerado para a realização do modelo. As outras partes, consideradas irrelevantes, são desconsideradas na montagem do modelo. Diversas áreas do conhecimento utilizam modelos para estudos do comportamento de seus componentes, como na engenharia, arquitetura, física, química e outros.

Para se trabalhar com modelos é necessário definir o meio no qual esse modelo será mostrado. Um exemplo simples é o modelo de um edifício. Pode ser um simples desenho técnico em papel, pode ser um desenho artístico colorido e em 3D, pode ser uma maquete em papelão ou outro material rígido, ou pode ser um conjunto de equações matemáticas de elementos finitos para simulação de esforços em computador. Então um modelo do edifício pode ser usado para mostrar a aparência física construtiva para fins de vendas ou parar determinar valores e quantidade dos materiais a serem utilizados na construção.

Um programa de computador tem o seu modelo feito em uma “linguagem de modelagem”. A UML (“Unified Modeling Language”) possui diversos conceitos de prototipagem de modelos. Os modelos podem utilizar imagens e textos para facilitar a compreensão da semântica e da notação do programa. O modelo é utilizado para facilitar o entendimento do processo a ser programado do que o próprio programa final.

3.2.1 – O uso de modelos

Os modelos são usados para diversos fins. Para cada aplicação ou finalidade, um tipo de modelo é utilizado. Um determinado projeto pode ter vários modelos, sendo cada qual destinado a solucionar um determinado problema. As principais aplicações de modelos são:

Mostrar, para as pessoas interessadas no projeto, as diversas partes constituintes, os requisitos precisos e o conhecimento aplicado, para entendimento e concordância. Os modelos utilizados para um edifício vão desde a aparência, tipo de tráfego, tipos de serviços comuns, disposição de mobiliário, até análises estruturais e resistências a ventos e terremotos. Outros modelos são também utilizados para determinar custos, materiais e construção. As pessoas interessadas podem ser o arquiteto, os engenheiros, o empreiteiro, os contratados, os clientes e a prefeitura da cidade.

Um sistema de programação pode adotar diferentes modelos para cada parte constituinte como o conhecimento da linguagem de programação, a interação com os usuários, a modularidade do programa, os padrões adotados e outros arranjos. As pessoas interessadas são aquelas ligadas ao desenvolvimento do sistema, como arquiteto de software, analistas e programadores. Também as pessoas ligadas à parte comercial, como os gerentes, clientes, financiadores, usuários e operadores.

Facilitar o entendimento do projeto de um sistema. Na arquitetura de um edifício são utilizados modelos em papel, com desenhos em 3D, com programas gráficos em computador e com equacionamento matemático para visualização e experimentação de diversas opções de projeto. As etapas de criação e modificação de modelos simples são essenciais para a criatividade e inovação e com baixo custo.

Um sistema de programação deve possuir um modelo simples e conciso que permite às pessoas envolvidas visualizarem as diversas soluções antes do código fonte ser escrito. A escolha de uma boa linguagem de modelagem permite ao projetista obter uma visão geral do modelo do processo ou sistema antes de fazer o detalhamento da programação.

Mostrar os detalhes do projeto de maneira a facilitar mudanças independente dos requisitos operacionais. Um cliente que vai adquirir uma unidade num edifício normalmente deseja ter uma visão ampla da aparência externa através de um modelo representativo, como uma maquete física ou virtual. As canalizações de água, esgoto, eletricidade e comunicações são apresentados em modelos para uso dos engenheiros da obra. A maquete do prédio é a versão final das muitas que o arquiteto considerou ser a melhor. Os clientes podem verificar as diversas informações do aspecto externo e não se preocupam com os detalhes construtivos internos.

Um modelo de um sistema de programação representa o comportamento de um processo ou sistema externo e as informações do mundo real que estão representadas no sistema. O comportamento externo do processo ou sistema é mostrado num modelo de classes e operações internas. O modelo final do comportamento do processo ou sistema mostra a abordagem que o projetista considerou razoável.

Gerar um produto utilizável. A maquete do edifício é uma parte da modelagem do projeto de um edifício. Outros produtos derivados podem ser a lista de materiais, visualização simulada, simulação de reação a ventos, cronograma de execução, lista de profissionais envolvidos e simulação de resistência da estrutura.

O modelo do sistema de software é usado para gerar outros produtos como declaração de classes, procedimentos operacionais, interface com usuário, banco de dados, configuração de parâmetros e condições de execução.

Gerenciar as informações relativas aos sistemas ou processo. Os modelos são utilizados para facilitar a organização das informações no sentido e localização, filtragem, recuperação, examinação e edição. Num modelo de um edifício, as informações estão organizadas por tipo de serviço, como projeto estrutural, projeto elétrico, projeto hidráulico e sanitário, projeto de ventilação, projeto de decoração e outros. Atualmente essas informações estão armazenadas em um computador com modelo em camadas. E isso facilita a localização e edição. Outrora os modelos eram representados em folhas de papel e as modificações eram trabalhosas por exigir redesenho de várias folhas. Com o modelo em computador, as diversas pessoas envolvidas no projeto podem compartilhar mais facilmente qualquer alteração ou proposta de alteração.

O modelo do sistema de programação organiza as informações em vários níveis para facilitar o entendimento de todo o sistema por parte de todas as pessoas envolvidas no desenvolvimento do programa. Assim, informações como a estrutura estática, estrutura funcional, interações, requisitos operacionais e funcionais, organização do banco de dados e apresentação são disponibilizadas na forma de modelos individuais como parte integrante do modelo completo.

Um protocolo e uma ferramenta de visualização e edição é fundamental para gerenciamento preciso de um modelo de qualquer tamanho. Editores gráficos podem gerenciar o modelo através de camadas de informação, que podem ser ocultadas se desnecessárias em uma visualização, formar grupos de entidades relacionadas e alterar propriedades em elementos individuais ou agrupados.

Avaliar as soluções econômicas. Os projetos de edifícios podem não ser, de início, totalmente esclarecidos com relação aos riscos e vantagens do empreendimento. As diversas partes do projeto podem não se inter-relacionar de um modo simples. Com a modelagem dos diversos projetos constituintes, o projetista consegue avaliar os custos financeiros e os riscos em cada etapa da implementação do projeto antes do início da construção do edifício.

A modelagem de um sistema de programação de grande porte permite a comparação de vários projetos propostos. Nos modelos propostos, o nível de detalhamento é baixo, mas o suficiente para mostrar a maioria dos problemas que o projeto final deve encontrar. Com uma boa modelagem, vários projetos podem ser considerados, e com uma comparação de custos proceder a uma escolha que atenda os requisitos propostos.

Simular sistemas complexos. Alguns eventos não são possíveis de avaliar em um mundo real. Por exemplo, um edifício sendo atingido por ventos fortes de um tornado. Um tornado real não pode ser produzido no mundo real e seus efeitos poderiam destruir os instrumentos de medição. Os modelos físicos de elementos complexos podem ser facilmente modelados e compreendidos. Os impactos de eventos físicos naturais como, ventos, chuva, neve, temperatura e insolação, podem ser aplicados aos modelos de edifícios e a análise dos resultados permitir mudanças no projeto para minimizar os efeitos sobre o projeto.

Em sistemas de programação, a modelagem permite entender a complexidade de vários subsistemas que não podem ser tratados diretamente. O modelo permite um nível de abstração ao nível humano sem aprofundamento em detalhes. Num modelo em computador podem ser realizadas análises complicadas para encontrar possíveis pontos de erros, como compatibilização de temporização e sobrecarga de recursos. Com um modelo é possível determinar resultados esperados de mudanças previstas antes que estas mudanças sejam implementadas. Também é possível determinar as mudanças necessárias no sistema de programação para reduzir os efeitos indesejados.

3.2.2 – Níveis de modelos

Os modelos são adaptados para cada aplicação e podem ter diversos níveis de abstração. O nível de detalhamento do modelo é ajustado para o tipo desejado de aplicação. As aplicações mais comuns são:

Guia para o processo de pensamento. São construídos modelos de alto nível no início de um projeto para direcionar o processo de pensamento e destacar as diversas opções de funcionalidade do projeto. Esses modelos são o ponto de partida do projeto do sistema pois carregam os requisitos operacionais necessários. Os modelos iniciais devem fornecer uma quantidade de opções para ajudar aos projetistas a escolher uma boa opção para o sistema conceitual. Os modelos iniciais, mais genéricos, vão sendo substituídos por modelos mais precisos, com mais detalhes. Eventualmente acontecem mudanças no processo de análise inicial do processo de pensamento e novos rumos são descobertos. Essas mudanças não precisam ser documentadas pois a estrutura do processo de pensamento é produzir ideias. Deve-se preservar os modelos de pensamento finais de cada foco analisado para evitar possíveis repetições. Os modelos iniciais não necessitam de detalhes ou precisão dimensional que são exigidos nos modelos de implantação do projeto. Também não necessitam de um conjunto completo de conceitos.

O modelo inicial é uma visão geral do sistema com precisão reduzida, normalmente um modelo de análise da atividade a ser realizada. Esse modelo deve ser preservado mesmo após o desenvolvimento passar para a próxima etapa de criação do projeto. As duas etapas mais importantes no processo de

evolução de um modelo é a adição de detalhes e a busca de elementos em sobreposição ou inconsistente. Raramente é necessário fazer um histórico de mudanças, apenas em situações especificadas pela equipe de criação e desenvolvimento do projeto.

Especificações abstratas da estrutura essencial de um sistema. Os principais conceitos e mecanismos do sistema são abordados nos modelos das fases de análise ou também do projeto preliminar. O sistema final vai incorporar estes conceitos e mecanismos totalmente ou com as modificações que forem propostas na evolução do projeto. Os detalhes do projeto são adicionados nas etapas futuras de evolução do projeto. Eventualmente alguns elementos podem ter modificações na etapa de execução do projeto, sendo necessária a alteração da documentação do projeto. Os modelos abstratos têm como objetivo a correção de possíveis problemas gerais e de alto nível antes da abordagem de detalhamento localizado. A evolução do modelo abstrato para o modelo final passa por uma análise cuidadosa que garante que os principais conceitos e mecanismos estejam presentes. Uma documentação precisa deve ser elaborada durante a execução dos modelos essenciais e dos modelos completos, tanto durante a fase de execução dos modelos quanto da fase de conclusão desses modelos. Se isso não for feito, não existe garantia que o sistema final vai incorporar os conceitos e propriedades que foram mostrados nos modelos essenciais. Os modelos essenciais são do tipo semântico e não incorporam todos os elementos da implementação final. A semântica lógica geralmente é obscurecida pelas distinções de desempenho de baixo nível. É comum que para fazer um modelo de implementação completo passe por um modelo essencial de forma clara e direta, e ele pode ser feito por um gerador automático de código ou desenvolvido por um programador.

Especificações completas de um sistema final. Um sistema de modelo de implementação deve ser construído com informações suficientes incluídas, como semântica lógica do sistema e algoritmos. Também deve ser incluído as estruturas de dados e mecanismos que garantem o desempenho adequado. Outras informações como decisões organizacionais sobre os artefatos do sistema que são necessários para o trabalho cooperativo por humanos e o processamento por ferramentas também são desadas que estejam incluídas. O objetivo é a compreensão humana e facilidades para programação em computador. Esses objetos não são parte do aplicativo final, mas são parte do processo de construção do modelo final.

Exemplos de sistemas típicos ou possíveis. Exemplos bem escolhidos podem fornecer informações aos humanos e validar as especificações e implementações do sistema. Mesmo uma grande coleção de exemplos, entretanto, necessariamente fica aquém de uma descrição definitiva. Em última análise, precisamos de modelos que especificam o caso geral; afinal, é isso que um programa é. No entanto, exemplos de estruturas de dados típicas, sequências de interação ou histórias de objetos podem ajudar um ser humano a tentar entender uma situação complicada. Os exemplos devem ser usados com algum cuidado. É logicamente impossível induzir o caso geral a partir de um conjunto de exemplos, mas protótipos bem escolhidos são a maneira como a maioria das pessoas pensa. Um modelo de exemplo inclui instâncias em vez de descritores gerais. Portanto, ele tende a ter uma sensação diferente de um modelo descritivo genérico. Os modelos de exemplo geralmente usam apenas um subconjunto de construções UML, aquelas que lidam com instâncias. Ambos os modelos descritivos e modelos exemplares são úteis na modelagem de um sistema.

Descrições completas ou parciais dos sistemas. Um modelo pode ser uma descrição completa de um único sistema sem referências externas. Mais frequentemente, é organizado como um conjunto de unidades distintas e discretas, cada uma das quais pode ser armazenada e manipulada separadamente como parte de toda a descrição. Esses modelos têm “pontas soltas” que devem ser ligadas a outros modelos em um sistema completo. Como as peças têm coerência e significado, elas podem ser combinadas com outras peças de várias maneiras para produzir muitos sistemas diferentes. Alcançar a reutilização é uma meta importante de uma boa modelagem.

Os modelos evoluem com o tempo. Modelos com maior grau de detalhes são derivados de modelos mais abstratos e modelos mais concretos são derivados de modelos mais lógicos. Por exemplo, um modelo pode começar como uma visão de alto nível de todo o sistema, com alguns serviços principais em breve detalhe e sem enfeites. Com o tempo, muito mais detalhes são adicionados e variações são introduzidas. Também com o tempo, o foco muda de uma visão lógica de front-end centrada no usuário para uma visão física de back-end centrada na implementação. À medida que os desenvolvedores trabalham com um sistema e o entendem melhor, o modelo deve ser iterado em todos os níveis para capturar esse entendimento; é impossível entender um grande sistema em uma única passagem linear. Não existe uma forma "certa" para um modelo.

3.2.3 – Conteúdo de um modelo

Semântica e apresentação. Os modelos têm dois aspectos principais: informação semântica (semântica) e apresentação visual (notação).

O aspecto semântico captura o significado de um aplicativo como uma rede de construções lógicas, como classes, associações, estados, casos de uso e mensagens. Os elementos do modelo semântico carregam o significado do modelo - ou seja, eles transmitem a semântica. Os elementos de modelagem semântica são usados para geração de código, verificação de validade, métricas de complexidade e assim por diante. A aparência visual é irrelevante para a maioria das ferramentas que processam modelos. A informação semântica é freqüentemente chamada de modelo. Um modelo semântico tem uma estrutura sintática, regras de boa formação e dinâmica de execução. Esses aspectos são frequentemente descritos separadamente (como nos documentos de definição da UML), mas são partes estreitamente inter-relacionadas de um único modelo coerente.

A apresentação visual mostra informações semânticas em uma forma que pode ser vista, pesquisada e editada por humanos. Os elementos de apresentação carregam a apresentação visual do modelo - ou seja, eles o mostram de uma forma diretamente apreensível por humanos. Eles não acrescentam significado, mas organizam a apresentação para enfatizar o arranjo do modelo de uma forma utilizável. Portanto, eles orientam a compreensão humana de um modelo. Os elementos de apresentação derivam sua semântica de elementos de modelo semântico. Mas, visto que o layout dos diagramas é fornecido por humanos, os elementos de apresentação não são completamente derivados de elementos lógicos. O arranjo dos elementos de apresentação pode transmitir conotações sobre relacionamentos semânticos que são muito fracos ou ambíguos para formalizar no modelo semântico, mas são, no entanto, sugestivos para os humanos.

Contexto. Os próprios modelos são artefatos em um sistema de computador e são usados em um contexto mais amplo que lhes dá todo o significado. Este contexto inclui a organização interna do modelo, anotações sobre o uso de cada modelo no processo geral de desenvolvimento, um conjunto de padrões e premissas para a criação e manipulação de elementos e uma relação com o ambiente em que são usados.

Os modelos requerem uma organização interna que permita o uso simultâneo por vários grupos de trabalho sem interferência indevida. Essa decomposição não é necessária por razões semânticas - um grande modelo monolítico seria tão preciso quanto um conjunto de modelos organizados em pacotes coerentes, talvez até mais preciso porque as fronteiras organizacionais complicam o trabalho de definir uma semântica precisa. Mas as equipes de trabalhadores não poderiam trabalhar de forma eficaz em um grande modelo monolítico sem ficar constantemente no caminho uns dos outros. Além disso, um modelo monolítico não possui peças que possam ser reaproveitadas em outras situações. Finalmente, as alterações em um modelo grande têm consequências difíceis de determinar. Mudanças em uma parte pequena e isolada de um modelo grande podem ser tratáveis se o modelo for adequadamente estruturado em subsistemas com interfaces bem definidas. Em qualquer caso, dividir grandes sistemas em uma

hierarquia de peças bem escolhidas é a maneira mais confiável de projetar grandes sistemas que os humanos inventaram ao longo de milhares de anos.

Os modelos capturam informações semânticas sobre um sistema de aplicativo, mas também precisam registrar muitos tipos de informações sobre o próprio processo de desenvolvimento, como o autor de uma classe, o status de depuração de um procedimento e quem tem permissão para editar um diagrama . Essas informações são, na melhor das hipóteses, periféricas à semântica do sistema, mas são importantes para o processo de desenvolvimento. Um modelo de sistema, portanto, precisa incluir ambos os pontos de vista. Isso é mais facilmente alcançado considerando as informações de gerenciamento de projetos como anotações ao modelo semântico - ou seja, descrições arbitrárias anexadas aos elementos do modelo, mas cujo significado está fora da linguagem de modelagem. Em UML, essas anotações são implementadas como strings de texto cujo uso é definido por perfis opcionais.

Os comandos usados para criar e modificar um modelo não fazem parte da semântica da linguagem de modelagem mais do que os comandos de um editor de texto ou navegador fazem parte da semântica de uma linguagem de programação. As propriedades do elemento de modelo não têm valores padrão; em um modelo específico, eles simplesmente têm valores. Para o desenvolvimento prático, entretanto, os humanos precisam construir e modificar modelos sem ter que especificar tudo em detalhes. Os valores padrão existem no limite entre a linguagem de modelagem e a ferramenta de edição que a suporta. Eles são realmente padrões nos comandos de ferramentas que criam um modelo, embora possam transcender uma ferramenta individual e se tornarem expectativas do usuário sobre a implementação da linguagem por ferramentas em geral.

Os modelos não são construídos e usados isoladamente. Eles fazem parte de um ambiente maior que inclui ferramentas de modelagem, linguagens e compiladores, sistemas operacionais, redes de computadores, restrições de implementação e assim por diante. As informações sobre um sistema incluem informações sobre todas as partes do ambiente. Algumas delas serão armazenadas em um modelo, embora não sejam informações semânticas. Os exemplos incluem anotações de gerenciamento de projeto (discutidas acima), dicas e diretivas de geração de código, pacote de modelo e configurações de comando padrão para uma ferramenta de editor.

Outras informações podem ser armazenadas separadamente. Os exemplos incluem o código-fonte do programa e comandos de configuração do sistema operacional. Mesmo que algumas informações façam parte de um modelo, a responsabilidade por interpretá-las pode estar em vários lugares, incluindo a linguagem de modelagem, a ferramenta de modelagem, o gerador de código, o compilador, uma linguagem de comando e assim por diante. Este livro descreve a interpretação dos modelos que são definidos na especificação UML e que, portanto, se aplica a todos os usos da UML. Mas, ao operar em um ambiente de desenvolvimento físico, outras fontes podem adicionar interpretações adicionais além das fornecidas na especificação UML.

3.2.4 – Significado de um modelo

Um modelo é um gerador de configurações potenciais de sistemas; os sistemas possíveis são sua extensão, ou valores. Idealmente, todas as configurações consistentes com o modelo devem ser possíveis. Às vezes, no entanto, não é possível representar todas as restrições dentro de um modelo. Um modelo também é uma descrição da estrutura genérica e do significado de um sistema. As descrições são sua intenção ou significado. Um modelo é sempre uma abstração em algum nível. Ele captura os aspectos essenciais de um sistema e ignora alguns dos detalhes. Os seguintes aspectos devem ser considerados para os modelos.

Abstração versus detalhe. Um modelo captura os aspectos essenciais de um sistema e ignora os outros. Quais são os essenciais é uma questão de julgamento que depende do propósito do modelo.

Esta não é uma dicotomia; pode haver um espectro de modelos de precisão crescente. Uma linguagem de modelagem não é uma linguagem de programação. Uma linguagem de modelagem pode permitir que os modelos sejam especificados em vários níveis de detalhe. Um modelo inicial ou um modelo de alto nível pode não exigir detalhes completos, porque detalhes adicionais podem ser irrelevantes para o propósito em questão. Modelos em diferentes níveis de precisão podem ser usados ao longo da vida de um projeto. Um modelo destinado à geração de código requer que, pelo menos, alguns problemas de linguagem de programação sejam resolvidos. Normalmente, os modelos têm baixa precisão durante a análise inicial. Eles ganham detalhes à medida que o ciclo de desenvolvimento avança, de modo que os modelos finais têm detalhes e precisão consideráveis.

Especificação versus implementação. Um modelo pode dizer o que algo faz (especificação) e também como a função é realizada (implementação). Esses aspectos devem ser separados na modelagem. É importante acertar o quê antes de investir muito tempo no como. A abstração da implementação é uma faceta importante da modelagem. Pode haver uma cadeia de vários relacionamentos de especificação-implementação, em que cada implementação define as especificações para a próxima camada.

Descrição versus instância. Modelos são descrições. As coisas que eles descrevem são instâncias, que geralmente aparecem em modelos apenas como exemplos. A maioria das instâncias existe apenas como parte da execução do tempo de execução. Às vezes, no entanto, as próprias instâncias de tempo de execução são descrições de outras coisas. Chamamos esses objetos híbridos de metadados.

Olhando mais profundamente, não é realista insistir que tudo é uma instância ou uma descrição. Algo é uma instância ou descrição não isoladamente, mas apenas em relação a outra coisa, e muitas coisas podem ser abordadas de vários pontos de vista.

Variações na interpretação. Existem muitas interpretações possíveis de modelos em uma linguagem de modelagem. Pode-se definir certos pontos de variação semântica - lugares em que diferentes interpretações são possíveis – e atribuir a cada interpretação um nome como uma variação semântica de modo que se possa afirmar qual variação está sendo usada.

Por exemplo, a linguagem Self tem um mecanismo diferente para encontrar métodos do que a linguagem Smalltalk; um ponto de variação semântica no mecanismo de resolução do método permite que qualquer linguagem de programação seja suportada. Os pontos de variação semântica permitem que diferentes modelos de execução sejam suportados.

3.2.5 – Estruturas de modelagem

Os diferentes diagramas que compõem a UML podem ser agrupados em categorias, levando em conta para isto o contexto em que cada uma dessas representações pode vir a ser empregada.

- **Diagramas Estruturais.** Priorizam a descrição estática de estruturas de um sistema, como classes, atributos e operações destas últimas, além de prováveis relacionamentos entre tais construções.
    - **Diagrama de Classes.** Permite a visualização de um conjunto de classes, detalhando atributos e operações (métodos) presentes nesta última, assim como prováveis relacionamentos entre essas estruturas. Este tipo de representação pode incluir ainda definições de interfaces.
    - **Diagrama de Componentes.** Apresenta diferentes componentes de um sistema, além de possíveis dependências entre tais elementos. A ideia de componente refere-se a uma parte (ou até mesmo um módulo) de uma aplicação, englobando assim uma séria de outras estruturas relacionadas (como classes, interfaces etc.).
    - **Diagrama de Pacotes.** Descreve as dependências entre diferentes namespaces/pacotes que compõem uma aplicação. Dentro da plataforma .NET, um namespace

costuma conter classes, interfaces e outros elementos, atuando como uma forma de agrupamento lógico destes elementos.

  - **Diagrama de Objetos.** Apresenta o estado de instâncias de objetos dentro de um sistema, levando em conta para isto um intervalo de tempo específico.
  - **Diagrama de Estrutura Composta.** Utilizado para demonstrar a estrutura interna de uma classe, incluindo referências que apontam para outras partes de um sistema.
  - **Diagrama de Instalação.** Empregado para demonstrar a estrutura de hardware adotada para a implantação de uma aplicação em um ambiente. Pode envolver dispositivos como servidores de aplicação, servidores de banco de dados, terminais de usuários etc.
  - **Diagrama de Perfil.** Possibilita a definição de novos elementos UML, permitindo assim estender os diagramas existentes com a inclusão de estruturas customizadas para uma determinada necessidade.

- **Diagramas Comportamentais.** Detalham o funcionamento (comportamento) de partes de um sistema ou processos de negócio relacionados a tal aplicação.
  - **Diagrama de Casos de Uso.** Voltado à apresentação de funcionalidades e características de um sistema, assim como de que forma tais elementos se relacionam com usuários e entidades externas envolvidas num determinado processo.
  - **Diagrama de Atividades.** Contempla as diversas tarefas desempenhadas na execução de uma atividade, sendo utilizado geralmente na representação de processos dentro de uma empresa/organização.
  - **Diagrama de Transição de Estados.** Detalha os diferentes estados pelos quais pode passar um objeto, tomando por base a execução de um processo dentro do sistema que se está considerando.

Diagramas de Interação. Considerados um subgrupo dos diagramas comportamentais, sendo normalmente utilizados na representação de interações entre objetos de uma aplicação.

  - **Diagrama de Sequência.** Demonstra as interações entre diferentes objetos na execução de uma operação, destacando ainda a ordem em que tais ações acontecem num intervalo de tempo. A sequência em que as diversas operações são executadas ocorre na vertical, de cima para baixo.
  - **Diagrama de Interatividade.** Espécie de representação híbrida, com uma estrutura similar à de diagramas de atividade. O que diferencia este tipo de representação está justamente no fato do equivalente a uma atividade ser representada por outro diagrama, sendo o de sequência um exemplo de uso válido neste último caso.
  - **Diagrama de Colaboração ou Comunicação.** Similar aos diagramas de sequência, é também empregado na modelagem de interações entre vários objetos dentro de um determinado contexto. Este tipo de representação difere de um diagrama de sequência por não possuir uma estrutura rígida para demonstrar a comunicação entre objetos, ou seja, estes elementos podem ser dispostos na melhor ordem que se julgar necessária, sem a obrigatoriedade de exibir as diferentes operações na vertical uma após a outra.
  - **Diagrama de Tempo.** Corresponde a um tipo específico de diagrama de sequência, descrevendo mudanças de estado e interações entre objetos dentro de intervalos de tempo tomados como parâmetro.

3.3 – Aplicação de modelagem na programação de CLPs

Como a maioria dos programas para controladores industriais são relativos ao funcionamento da máquina ou processo, os modelos comportamentais e de interação são os mais utilizados. Será abordado apenas alguns modelos com aplicações no funcionamento e operação automática de máquinas e processos industriais.

3.3.1 – Sequência de passos

As atividades da máquina ou processo são descritas num formalismo textual e agrupadas numa ordem de execução das tarefas. O conjunto de tarefas em execução tem uma única condição de transição e a transferência de atividades é feita para o conjunto de tarefas imediatamente posterior.

Na figura 3.2 pode ser visto um exemplo de um controle de abertura de uma porta, onde "Abrir" é um botão de comando para abrir a porta, "Fechar" é um botão de comando para fechar a porta, "Aberto" é um sensor que é ativo quando a porta está aberta, "Fechado" é um sensor que é ativo quando a porta está fechada, "Ma" é a etiqueta do relé para fazer girar o motor para abrir a porta e "Mf" é a etiqueta do relé para fazer girar o motor para fechar a porta.

Passo 01	Nenhuma ação. Se (Abrir = 1) então vá para o Passo 2.
Passo 02	Ligar motor para abrir (Ma = 1). Se (Aberto = 1) então vá para o Passo 3.
Passo 03	Nenhuma ação. Se (Fechar = 1) então vá para o Passo 4.
Passo 04	Ligar motor para fechar (Mf = 1). Se (Fechado = 1) então vá para o Passo 1.

Figura 3.2 – Exemplo de uma modelagem por sequência de passos.

3.3.2 – Diagrama de tempos

Semelhante ao modo sequência de passos, com a diferença que todas as transições de condição operacional da máquina ou processo é feita por término de um intervalo de tempo. Esse foi um dos primeiros "controladores de processo" utilizados na indústria devido a sua construção inicialmente mecânica e depender exclusivamente de uma base de tempo constante (relógio).

Na figura 3.2 pode ser visto um exemplo de um controle de um processo industrial de um tanque de mistura de líquidos, composto por 2 válvulas de entrada de produto (Válvula 1 e Válvula2), uma válvula de saída de produto (Válvula 3) e um agitador com motor (Agitador). Os instantes de ligar e desligar cada dispositivo de saída é colocado num gráfico de tempos.

A "Válvula 1" é ligada no tempo "zero" e no tempo de 35 minutos e desligada no tempo 20 minutos e 40 minutos. A "Válvula 2" é ligada no tempo de 5 minutos e 40 minutos e desligada no tempo de 20 minutos e 45 minutos. A "Válvula 3" é ligada no tempo de 25 minutos e 45 minutos e desligada no tempo de 35 minutos e 50 minutos. O "Agitador" é ligado no tempo de 5 minutos e 40 minutos e desligado no tempo de 20 minutos e 45 minutos.

Para cada evento de tempo de ligar e desligar é atribuído um temporizador, que, neste exemplo, são 8 temporizadores identificados de T1 até T8, com os tempos especificados no retângulo tracejado.

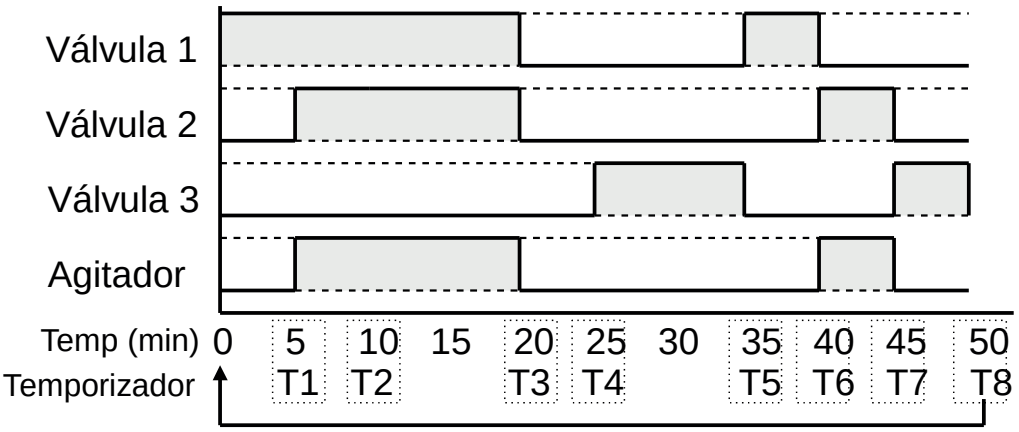

Figura 3.3 – Exemplo de uma modelagem por diagrama de tempos.

3.3.3 – Fluxograma

Cada atividade da máquina ou processo é colocado numa estrutura gráfica específica de representa o tipo de função executada. É possível especificar novos símbolos gráficos para cada grupo de tarefas associados aos símbolos padrão já concebidos. Os diversos símbolos permitem agrupar tarefas de execução, espera, decisão, repetição e duplicidade de ações.

Na figura 3.4 pode ser visto um exemplo de um controle cilindro pneumático que é acionado por uma válvula 5/2 vias de duplo solenoide. O solenoide com etiqueta “A+” é para o avanço do êmbolo do cilindro e o solenoide com etiqueta “A-” é para o recuo do embolo do cilindro. O sensor com etiqueta “A1” detecta o final do avanço do êmbolo e o sensor com etiqueta “A0” detecta o final do recuo do êmbolo. O botão com etiqueta “Ba” é para ativar o avanço do êmbolo do cilindro e o botão com a etiqueta “Br” é para ativar o recuo do êmbolo do cilindro.

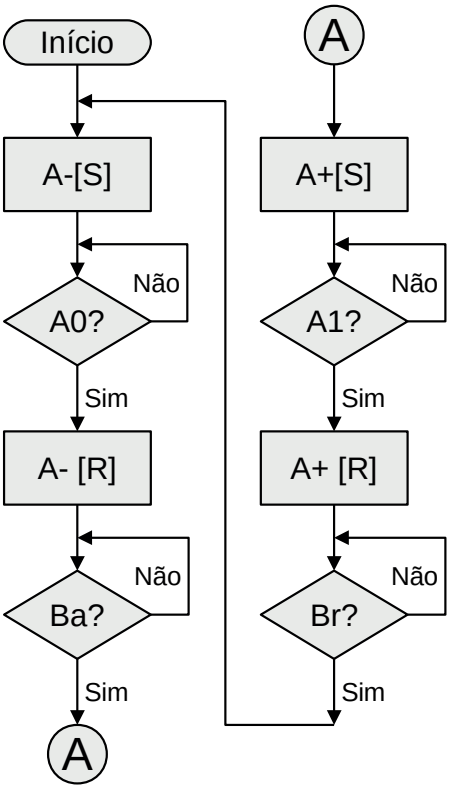

Figura 3.4 – Exemplo de uma modelagem por fluxograma.

3.3.4 – Diagrama de estados

É um diagrama simples com apenas dois tipos de símbolos onde o símbolo de estado aloca as ações associadas ao este estado e o símbolo de transição que aloca a lógica associada, que quando verdadeiro, permite a evolução de um estado para outro.

Na figura 3.5 pode ser visto um exemplo de um somador de moedas de 25 centavos e 50 centavos para totalizar 100 centavos. As moedas de 25 centavos ativam o sensor “M25” e as moedas de 50 centavos ativam o sensor “M50”. O valor total é armazenado na variável interna “V”.

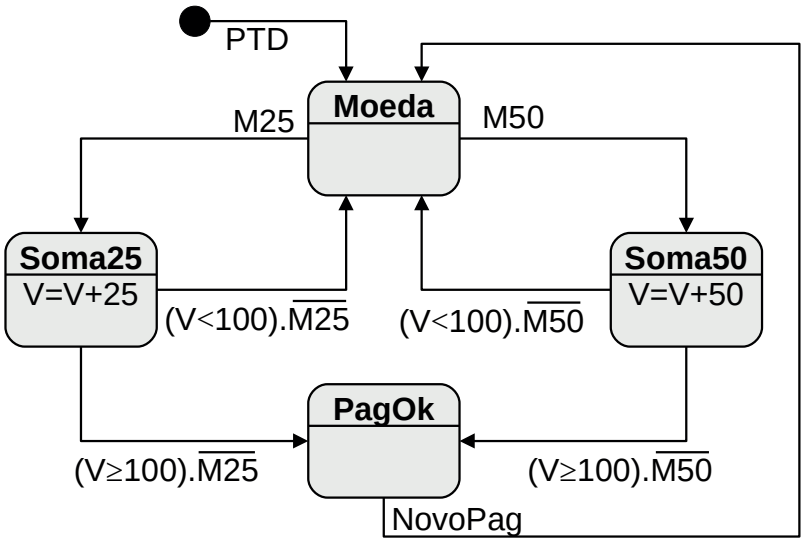

Figura 3.5 – Exemplo de uma modelagem por diagrama de estados.

3.3.5 – Gráfico de função sequencial

É uma estrutura mais complexa e rígida, com permissividade de execução de tarefas em paralelo e tarefas consecutivas. Possui facilidades de níveis de compreensão e facilidades computacionais, tornando esse método facilmente implementado em computador como a própria linguagem de programação.

Na figura 3.6 pode ser visto um exemplo de um dosador e misturador de produtos. O processo tem início quando é pressionado um botão de ligar (PTD). As balanças BM1 e BM2 funcionam em paralelo, de maneira semelhante: primeiro, é despejado o material de um dos silos até que seja disparado o sinal correspondente ao peso desejado do produto; em sequência, é despejado o material do segundo produto até que se obtenha um outro sinal correspondente ao peso desejado do produto. Após ser feita a dosagem de cada um dos produtos, as válvulas de saída das balanças (VBM1 e VBM2) devem ser abertas por 10 segundos para o seu esvaziamento no tanque de mistura. Para realizar uma nova dosagem, o botão de ligar deve ser acionado novamente.

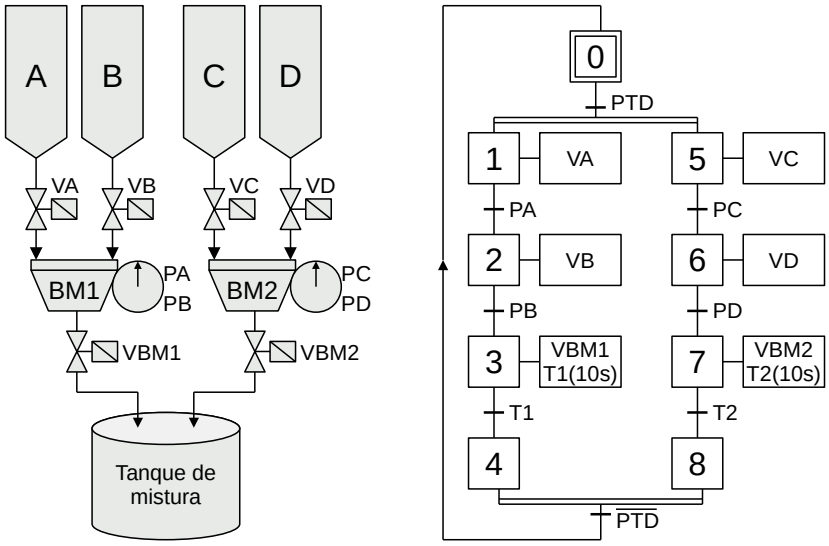

Figura 3.6 – Exemplo de uma modelagem por gráfico de função sequencial.

3.3.6 – Guia de marchas e paradas

Elaborado para facilitar a programação de tarefas normalmente encontradas no ambiente industrial. É um diagrama estrutural com detalhamento de componentes e características de evolução entre componentes.

3.4 – Conclusão

A programação estruturada é um paradigma de programação que visa melhorar a clareza, a qualidade e o tempo de desenvolvimento de um programa de computador, fazendo uso extensivo das construções de fluxo de controle estruturado de seleção e repetição, estruturas de bloco e sub-rotinas em contraste com o uso de testes e saltos simples.

Um dos conceitos mais importantes de programação é a capacidade de controlar um programa de forma que diferentes linhas de código sejam executadas ou que algumas linhas de código sejam executadas várias vezes. Os mecanismos que nos permitem controlar o fluxo de execução são chamados de estruturas de controle. O fluxograma é um método de documentar o fluxo que um programa executaria. Existem três categorias principais de estruturas de controle:

Sequência – muito chata. Simplesmente execute uma instrução, depois a próxima e a próxima. Basta executá-los em uma determinada sequência ou na ordem listada. A maioria das linhas de código são assim.

Seleção – aqui é onde você seleciona ou escolhe entre dois ou mais fluxos. A escolha é decidida fazendo-se algum tipo de pergunta. A resposta determina o caminho (ou quais linhas de código) serão executadas.

Iteração – Também conhecida como repetição, permite que algum código (de uma a várias linhas) seja executado (ou repetido) várias vezes. O código pode não ser executado (repetir zero vezes), executado um número fixo de vezes ou executado indefinidamente até que alguma condição seja atendida. Também conhecido como laço (“looping”) porque o fluxograma mostra o laço de volta para repetir a tarefa.

Uma quarta categoria descreve o código não estruturado.

Ramificação – Uma estrutura não controlada que permite que o fluxo de execução pule para uma parte diferente do programa. Esta categoria raramente é usada na programação estruturada modular.

Todas as linguagens de programação de alto nível têm estruturas de controle. Todas as linguagens têm as três primeiras categorias de estruturas de controle (sequência, seleção e iteração). A maioria tem a estrutura "if-then-else" (que pertence à categoria de seleção) e a estrutura "while" (que pertence à categoria de iteração). Após essas duas estruturas básicas, geralmente existem variações de linguagem.

Termos chave

Ramificação

Uma estrutura não controlada que permite que o fluxo de execução salte para uma parte diferente do programa.

Estruturas de controle

Mecanismos que nos permitem controlar o fluxo de execução de um programa.

Iteração

Uma estrutura de controle que permite que algumas linhas de código sejam executadas várias vezes.

Seleção

Uma estrutura de controle onde o programa escolhe entre duas ou mais opções.

Sequência

Uma estrutura de controle onde o programa executa os itens na ordem listada.

Código espaguete

Uma frase pejorativa para código-fonte não estruturado e difícil de manter.

O programa estruturado consiste em módulos bem estruturados e separados. Mas a entrada e saída em um programa estruturado é um evento único. Isso significa que o programa usa elementos de entrada única e saída única. Portanto, um programa estruturado é um programa bem mantido, organizado e limpo. Esta é a razão pela qual a abordagem de programação estruturada é bem aceita no mundo da programação.

Vantagens da abordagem de programação estruturada:

- Mais fácil de ler e entender.
- Amigo do usuário.
- Mais fácil de manter.
- Principalmente baseado em problemas, em vez de ser baseado em máquinas.
- O desenvolvimento é mais fácil porque requer menos esforço e tempo.
- Mais fácil de depurar.
- Independente de máquina, principalmente.

Desvantagens da abordagem de programação estruturada:

- Por ser independente da máquina, leva tempo para ser convertido em código de máquina.
- O código de máquina convertido não é o mesmo da linguagem "assembly".
- O programa depende de fatores mutáveis, como tipos de dados. Portanto, ele precisa ser atualizado com a necessidade em movimento.
- Normalmente, o desenvolvimento desta abordagem leva mais tempo, pois depende do idioma. Já no caso da linguagem "assembly", o desenvolvimento leva menos tempo, pois é fixo para a máquina.

Capítulo 4
Estrutura de programação por sequência de passos

4.1 – Introdução

Este método consiste na observação da máquina e identificação dos movimentos dos dispositivos atuadores e das ações executadas pelo controlador. Essa identificação de eventos deve ser colocada numa ordem de execução sequencial onde é identificado o início e final de cada evento.

O funcionamento da máquina ou processo é analisado e coletado informações sobre os elementos de atuação que são afetados por elementos sensores. É atribuído um início para o processo e a partir disso todos as ações dos atuadores são dependentes de uma ação anterior. Eventualmente o processo se repete e então volta-se ao início.

Enquanto que no método de solução por álgebra booleana, cada linha de programa é um programa único que associa uma saída a uma ou mais entradas, neste método, várias linhas de programa representam um mesmo programa, sendo um grupo de linhas com registradores de memória e um grupo de linhas com a lógica de saída para os atuadores externos. O sistema se assemelha a uma máquina de estados em configuração Moore.

Este método é aplicável somente a processos sequenciais simples e que não possuem estrutura de decisão para alteração do modo operacional da máquina. Uma alternativa para aplicação deste método em estruturas com decisão de escolha de sequências é a confecção de um programa de sequência de passos para cada sequência operacional da máquina e fazer os programas se interligarem por variáveis ou por passos de espera.

A maioria dos processos industriais são sequenciais simples, repetitivos e sem decisões de escolhas operacionais. Isso faz com que este método simples seja facilmente aplicado na solução dos problemas de automação industrial.

4.2 – Descrição do método

Para aplicar este método use os seguintes procedimentos:

1. **Entenda o processo.** Consiste em analisar todos os dispositivos de atuação e verificar qual é atuado primeiro, qual é o próximo atuador ser ativado, quais as fases de espera e quais os possíveis problemas de funcionamento.
2. **Escreva os passos da operação na sequência e atribua um número a cada um.** Esta sequência pode ser do tipo: 1, 2, 3, … Ou pode ser em intervalos de 10 para permitir inserção de novas linhas após a edição do programa. Fica assim: 10, 20, 30, … É possível a divisão da operação da máquina ou processo em vários “programas”. Neste caso atribua uma letra a um programa e outras letras aos outros programas. Exemplo: A1, A2, A3, … B1, B2, B3, …
3. **Para cada passo atribua uma ação.** Essa ação está diretamente relacionada com um dispositivo de saída ou uma variável interna. É possível a atribuição de nenhuma ação caso a máquina ou processo esteja em um modo de espera. Esta espera está relacionada com alguma entrada externa, sensor ou botoeira, ou interna através de um temporizador ou contador ou outro registrador. Identifique, também, o evento que encerra esta ação, normalmente associado a algum dispositivo de entrada. Uma maneira simples de

representação é fazer uma frase de texto que relacione a ação que está acontecendo com a finalização desta ação.

Passo N. A saída X tem a ação Y **até** que o evento Z aconteça.

4. **Faça a "modelagem funcional" da máquina ou processo.** Utilize de uma estrutura de texto onde possa aparecer as três informações importantes: número do passo, ação associada e lógica de finalização da ação. Uma sugestão pode ser vista na Figura 4.1.

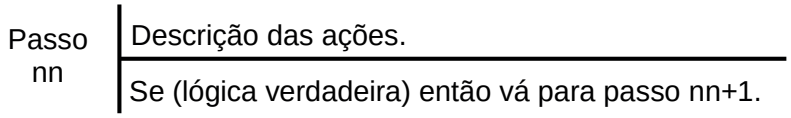

Figura 4.1 – Estrutura funcional para sequência de passos.

5. **Escreva a lógica Ladder para ativar e desativar os passos assim que o processo move-se através dos passos.** Ou seja, cada passo é uma memória ou bobina interna ativado ("Set") quando o passo correspondente for verdadeiro e desativado ("Reset") quando a lógica for verdadeira e enviar o controle para o próximo passo. Uma sugestão pode ser vista na Figura 4.2. Para executar esse processo de ativação e desativação de passos é necessário que um passo inicial esteja ativado. Normalmente é o passo número 1 e esta ativação é feita por um dispositivo de "primeiro ciclo" (em inglês: first scan). Existem algumas maneiras de fazer isso.

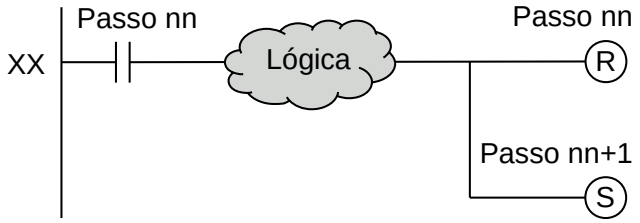

Figura 4.2 – Desativação e ativação de um passo na sequência de passos.

6. **Escreva a lógica Ladder para atribuir uma função da máquina para cada passo.** Se um passo tem uma ação de um dispositivo atuador ou dispositivo interno (temporizador, contador ou outro registrador), uma linha de programa é construído com um contato do passo associado e uma bobina do dispositivo ativado. Se mais de um passo ativa o mesmo dispositivo, o contato correspondente a este outro passo fica em paralelo com o contato anterior. Uma sugestão pode ser vista na Figura 4.3.

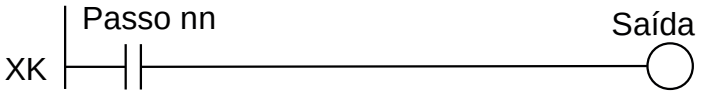

Figura 4.3 – Ativação de um dispositivo de saída.

7. Se o processo é repetitivo, faça o último passo retornar ao primeiro passo.

4.3 – Sistemas simples

Máquinas simples possuem apenas um único modo operacional, executando sempre as mesmas tarefas. A sequência de passos é única e simples e envolve apenas os elementos de entrada e de saída.

4.3.1 – Exemplo 1. O problema de subir e descer uma bandeira

Considere o problema de projetar um controlador para subir e descer uma bandeira.

Descrição da máquina ou processo

Existe um mastro com um cabo de aço preso em duas roldanas. No cabo de aço existem presilhas para prender uma bandeira. A roldana inferior possui um motor de indução reversível acoplado através de um sistema de engrenagens de redução de velocidade. Existem dois relés para acionamento do motor. Um dos relés, quando ativado, permite que o motor gire num sentido tal que faça a bandeira subir. O outro relé, quando ativado, permite que o motor gire num sentido tal que faça a bandeira descer. Os dois relés não podem ser ativados simultaneamente. Existem dois sensores de posição que detectam quando a bandeira está embaixo e quando a bandeira está em cima. Existe um painel de controle com uma chave seletora giratória de duas posições para ligar e desligar o controlador, uma botoeira para dar partida ao programa, uma lâmpada sinalizadora indicando que o programa está funcionando, uma botoeira para subir a bandeira, uma lâmpada sinalizadora indicando que o motor está funcionando para subir a bandeira, uma botoeira para abaixar a bandeira e uma lâmpada sinalizadora indicando que o motor está funcionando para abaixar a bandeira. Não existe solicitação para parada da bandeira a "meio-mastro". Um esquema do processo pode ser visto na Figura 4.4.

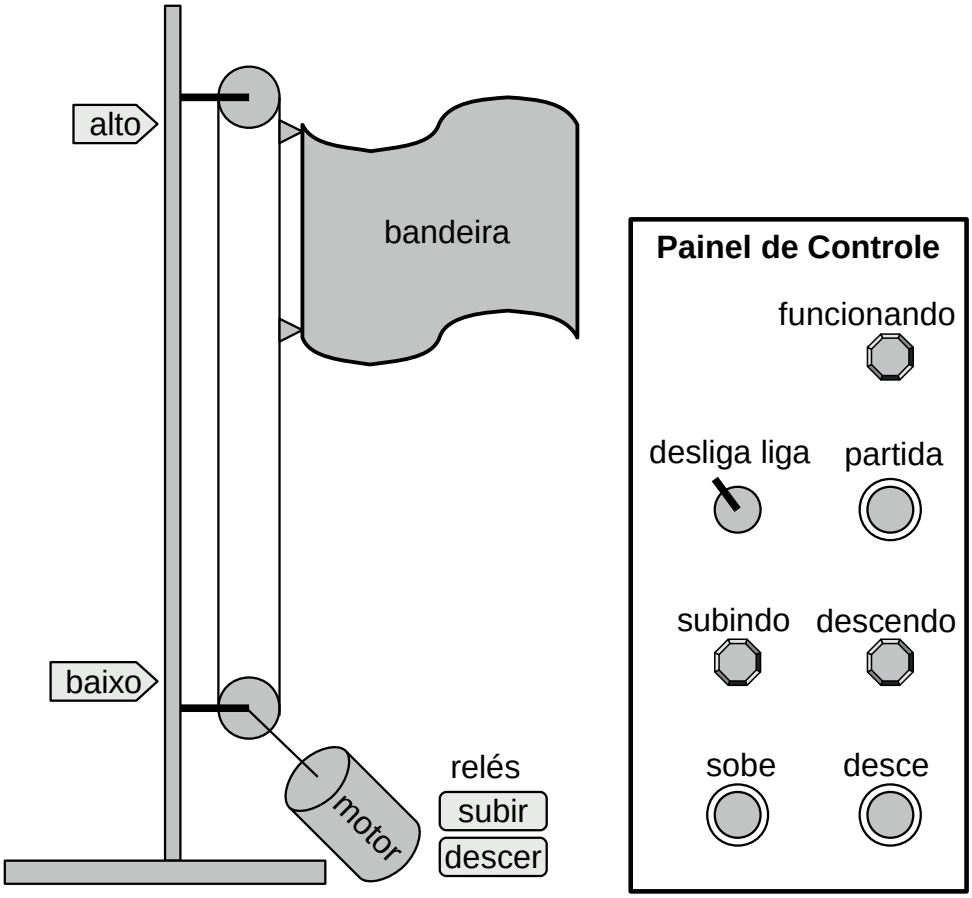

Figura 4.4 – Problema de automação da bandeira.

Descrição operacional da máquina ou processo

A bandeira deve subir quando o botão de subida for pressionado e descer quando o botão de descida for pressionado. Quando o sensor que está encima, no mastro, for acionado a bandeira deve parar. Quando o sensor que está embaixo, no mastro, for acionado a bandeira deve parar. Quando o equipamento for ligado, a bandeira deve descer até embaixo do mastro. Os botões para subir e para descer a bandeira devem ser ignorados durante o movimento da bandeira.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. Observe que a chave "desliga / liga" não está conectada a nenhum pino de entrada do controlador. A chave serve para fornecer energia elétrica para o controlador e os dispositivos de saída. A tabela das entradas pode ser vista na Tabela 4.1.

Parafuso	Etiqueta	Descrição
I01	alto	Sensor de fim-de-curso.
I02	baixo	Sensor de fim-de-curso.
I03	partida	Botoeira NA.
I04	sobe	Botoeira NA.
I05	desce	Botoeira NA.

Tabela 4.1 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na Tabela 4.2.

Parafuso	Etiqueta	Descrição
Q01	subir	Relé de acionamento do motor.
Q02	descer	Relé de acionamento do motor.
Q03	funcionando	Lâmpada sinalizadora de painel.
Q04	subindo	Lâmpada sinalizadora de painel.
Q05	descendo	Lâmpada sinalizadora de painel.

Tabela 4.2 – Descrição das saídas.

Solução por sequência de passos

Na análise do funcionamento da máquina pode ser visto a existência de eventos bem distintos: o movimento da bandeira para cima, o movimento da bandeira para baixo, a bandeira fica parada no alto do mastro e a bandeira fica parada no baixo do mastro. Esses eventos podem ser associados a ações sobre atuadores (relés do motor) e serem associados a comandos externos (botoeiras para subir e descer).

Colocando esses eventos em uma ordem sequencial e dentro do formato de uma frase condicional, é possível uma simulação do encadeamento das ações e sensores associados à máquina. As frases são então construídas utilizando a descrição da ação e não a etiqueta do dispositivo associado.

Os passos de operação da máquina podem ser assim identificados:

Passo 1. A bandeira move-se para baixo **até** encontrar o sensor de limite inferior.

Passo 2. A bandeira espera no baixo do mastro **até** o botão de subida ser pressionado.

Passo 3. A bandeira move-se para cima **até** encontrar o sensor de limite superior.

Passo 4. A bandeira espera no alto do mastro **até** o botão de descida ser pressionado.

Depois das frases condicionais montadas, procede-se à montagem da modelagem funcional, utilizando um formato de texto simples, conforme mostrado na figura 4.1.

A descrição das ações pode ser feita nos seguintes modos.

1. Descrição textual: "Ativar a saída digital X."; "Ativar e memorizar a saída digital Y."; "Desativar a saída memorizada Z."
2. Descrição por representação booleana: X; Y[S]; Z[R]

Para implementar o passo 1 verifica-se que para a bandeira mover-se para baixo é necessário ligar o relé "descer". Isso ativará o motor e o cabo de aço se moverá no sentido de descer a bandeira. Quando o marcador de bandeira tocar no sensor de limite inferior etiquetado como "embaixo" o relé "descer" deve ser desligado. Essa estrutura pode ser visto na Figura 4.5.

Passo 1	Ativar as saídas "descer" e "descendo".
	Se ("baixo" é verdadeiro) então vá para o Passo 2.

Figura 4.5 – Estrutura de passos para o Passo 1.

Para implementar o passo 2 verifica-se que o controlador não executa nenhuma ação sobre os relés de saída e fica esperando que o operador humano aperte o botão "sobe". Essa estrutura pode ser visto na Figura 4.6.

Passo 2	Nenhuma ação.
	Se ("sobe" é verdadeiro) então vá para o Passo 3.

Figura 4.6 – Estrutura de passos para o Passo 2.

Para implementar o passo 3 verifica-se que para a bandeira mover-se para cima é necessário ligar o relé "subir". Isso ativará o motor e o cabo de aço se moverá no sentido de subir a bandeira. Quando o marcador de bandeira tocar no sensor de limite superior etiquetado como "encima" o relé "subir" deve ser desligado. Essa estrutura pode ser visto na Figura 4.7.

Passo 3	Ativar as saídas "subir" e "subindo".
	Se ("alto" é verdadeiro) então vá para o Passo 4.

Figura 4.7 – Estrutura de passos para o Passo 3.

Para implementar o passo 4 verifica-se que o controlador não executa nenhuma ação sobre os relés de saída e fica esperando que o operador humano aperte o botão "desce". Observe que o processo se repete e o próximo passo é o passo inicial, ou seja, passo 1. Essa estrutura pode ser visto na Figura 4.8.

Passo 4	Nenhuma ação.
	Se (“desce” é verdadeiro) então vá para o Passo 1.

Figura 4.8 – Estrutura de passos para o Passo 4.

Codificação da sequência de passos na linguagem Ladder

Para efetuar a montagens das linhas de programação em linguagem Ladder, é sugerido a divisão das linhas de programação em 3 regiões distintas:

1. **Região de inicialização.** Conjunto de linhas de programação de preparação e inicialização da estrutura de passos.
2. **Região de lógica.** Conjunto de linhas de programação que implementa a lógica de transferência de validação de um passo para outro passo. É a parte de baixo da estrutura de passos.
3. **Região de saída.** Conjunto de linhas de programação que associa os passos às saídas.

De maneira gráfica, essa divisão de linhas de programação pode ser visto na Figura 4.9.

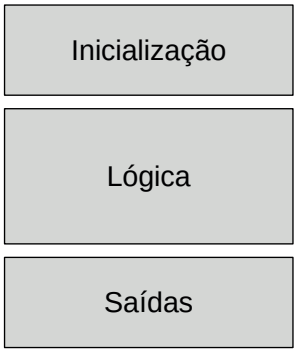

Figura 4.9 – Divisão das regiões das linhas de programação na estrutura de passos.

Na seção de inicialização são colocadas as linhas de programação em linguagem Ladder que permitem a partida do programa, ou seja, ativação do passo inicial. No nosso caso vamos utilizar a botoeira “partida” para ativar o passo número 1. Isso deve ser feito de tal maneira que o operador humano execute essa tarefa apenas uma única vez, desabilitando a botoeira “partida” assim que a estrutura de passos estiver funcionando. Normalmente as memórias internas, ou bobinas temporárias, estão no estado de falso sempre que o controlador é ligado e então não é necessário fazer o desligamento (Reset) dessas memórias.

Na Figura 4.10 pode-se ver a linha de programa número 1 que é o ativador do passo 1 (Set) através da botoeira “partida”. Observe que o contato “Passos” está em negação de falso, ou seja, está em verdadeiro. Também é ativada a memória “Passos” que desabilita a botoeira “partida” após a execução do ciclo de programa.

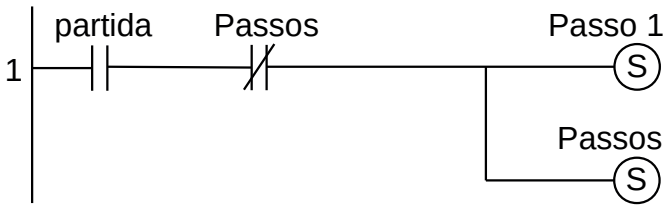

Figura 4.10 – Inicialização do programa.

Na seção de lógica são colocadas as linhas de programação em linguagem Ladder que permitem a transferência de validade de um passo para outro passo. Observe que apenas um único passo deve estar ativo de cada vez e que para transferir de um passo para outro passo é necessário a validação de uma condição de lógica.

Na Figura 4.11 pode-se ver a implementação do “Passo 1”. Nesta linha de programa a variável “Passo 1” está ativa, ou verdadeira, pela linha de programa anterior. Quando a entrada digital “baixo” for verdadeira, a variável “Passo 1” torna-se falsa (“Reset”) e a variável “Passo 2” torna-se verdadeira (“Set”).

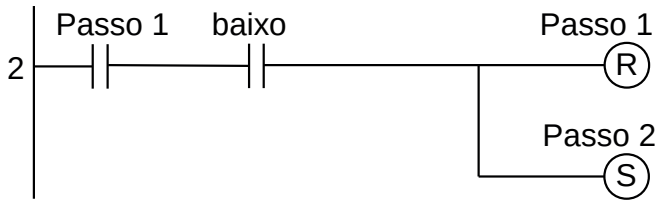

Figura 4.11 – Lógica para o Passo 1.

Na Figura 4.12 pode-se ver a implementação do “Passo 2”. Nesta linha de programa a variável “Passo 2” está ativa, ou verdadeira, pela linha anterior. Quando a entrada digital “sobe” for verdadeira, a variável “Passo 2” torna-se falsa (“Reset”) e a variável “Passo 3” torna-se verdadeira (“Set”).

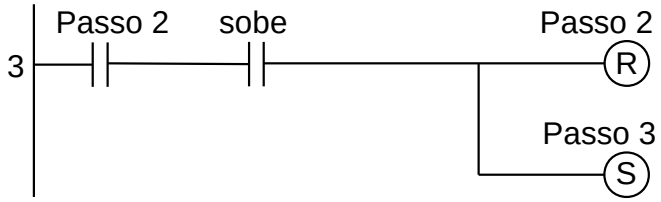

Figura 4.12 – Lógica para o Passo 2.

Na Figura 4.13 pode-se ver a implementação do “Passo 3”. Nesta linha de programa a variável “Passo 3” está ativa, ou verdadeira, pela linha anterior. Quando a entrada digital “alto” for verdadeira, a variável “Passo 3” torna-se falsa (“Reset”) e a variável “Passo 4” torna-se verdadeira (“Set”).

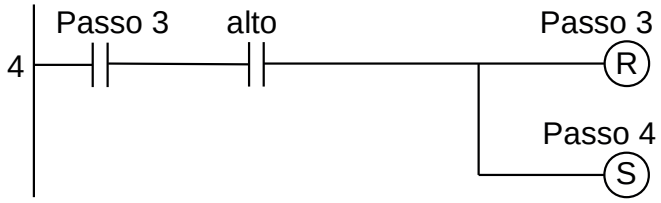

Figura 4.13 – Lógica para o Passo 3.

Na Figura 4.14 pode-se ver a implementação do “Passo 4”. Nesta linha de programa a variável “Passo 4” está ativa, ou verdadeira, pela linha anterior. Quando a entrada digital “desce” for verdadeira, a variável “Passo 4” torna-se falsa (“Reset”) e a variável “Passo 1” torna-se verdadeira (“Set”).

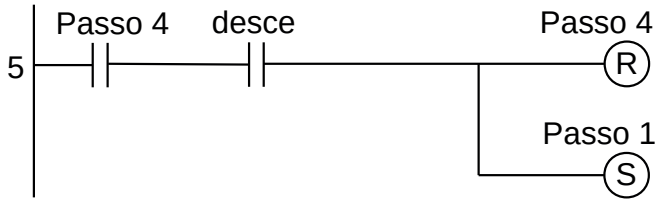

Figura 4.14 – Lógica para o Passo 4.

Na seção de saída são colocadas as linhas de programação em linguagem Ladder que permitem a associação de um ou mais passos com uma determinada saída. Todas as saídas desse programa podem ser vistas na Figura 4.15.

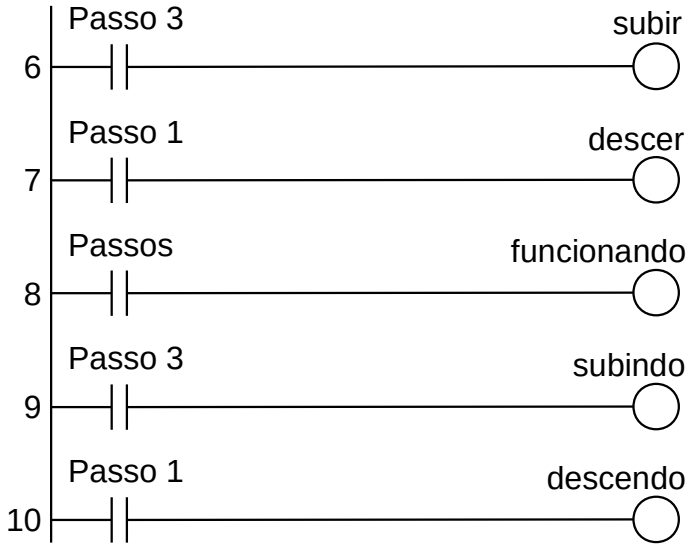

Figura 4.15 – Ativação das saídas.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 4.1 e 4.2. A tabela com as variáveis internas pode ser vista na Tabela 4.3.

Variável	Etiqueta	Descrição
M1	Passo 1	Memória para o passo 1.
M2	Passo 2	Memória para o passo 2.
M3	Passo 3	Memória para o passo 3.
M4	Passo 4	Memória para o passo 4.
M5	Passos	Memória para o programa em funcionamento.

Tabela 4.3 – Descrição das variáveis internas.

4.3.2 – Exemplo 2. O problema do botão e da lâmpada

Este problema foi apresentado em um capítulo anterior e foi visto dois modos de solução. O processo é simples e sequencial, então usar o método de sequência de passos permite compreender o funcionamento da máquina de um modo mais natural.

Descrição da máquina ou processo

Uma botoeira, de contato momentâneo, é conectada a uma entrada digital de um controlador.

Uma lâmpada é conectada a uma saída digital deste mesmo controlador.

Descrição operacional da máquina ou processo

O funcionamento é o seguinte:

1) O operador humano aperta o botão e a lâmpada acende imediatamente.

2) O operador humano continua com o dedo no botão e a lâmpada continua acesa.
3) O operador humano retira o dedo do botão e a lâmpada continua acesa.
4) O operador humano aperta, novamente, o botão e a lâmpada apaga imediatamente.
5) O operador humano continua com o dedo no botão e a lâmpada continua apagada.
6) O operador humano retira o dedo do botão e a lâmpada continua apagada.

As limitações para a solução deste problema são:

a) O programa deve utilizar os comandos básicos da linguagem "Ladder": contato normal, contato inversor e bobina normal ou com memória.
b) A quantidade de linhas de programação deve ser a menor possível.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na Tabela 4.4.

Parafuso	Etiqueta	Descrição
I01	botão	Botoeira NA.

Tabela 4.4 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na Tabela 4.5.

Parafuso	Etiqueta	Descrição
Q01	lâmpada	Lâmpada comum.

Tabela 4.5 – Descrição das saídas.

Solução por sequência de passos

O processo apresenta as seguintes situações facilmente identificadas: lâmpada apagada, lâmpada acesa, lâmpada acesa e com o botão pressionado, lâmpada apagada e o com o botão pressionado.

Essas situações devem ser enumeradas e sequenciadas, identificando os eventos de entrada e de saída.

Os passos de operação da máquina podem ser assim identificados:

Passo 1. A lâmpada está apagada **até** o operador humano pressionar o botão.

Passo 2. A lâmpada está acesa **até** o operador humano deixar de pressionar o botão.

Passo 3. A lâmpada está acesa **até** o operador humano pressionar o botão.

Passo 4. A lâmpada está apagada **até** o operador humano deixar de pressionar o botão.

A sequência de passos pode ser montada de acordo com a Figura 4.16.

Passo 1	Nenhuma ação. Se (“botão” é verdadeiro) então vá para o Passo 2.
Passo 2	Ativar a saída “lâmpada”. Se (“botão” é falso) então vá para o Passo 3.
Passo 3	Ativar a saída “lâmpada”. Se (“botão” é verdadeiro) então vá para o Passo 4.
Passo 4	Nenhuma ação. Se (“botão” é falso) então vá para o Passo 1.

Figura 4.16 – Estrutura de passos para o problema do botão e da lâmpada.

Codificação da sequência de passos na linguagem Ladder

Devido ao fato de existir apenas um único botão, não é possível utilizar a solução da Figura 4.10 para inicialização do programa. Muitos controladores incorporam memórias internas de uso especial. Uma dessas memórias é chamada de “first scan” que é um contato fechado, ou verdadeiro, no primeiro ciclo de programa e fica permanentemente aberto, ou falso, nos outros ciclos de programa. A maioria dos fabricantes adota a simbologia “f.s.” para este contato. Essa estrutura pode ser vista na Figura 4.17.

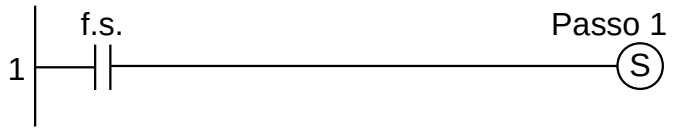

Figura 4.17 – Inicialização do programa com contato de “first scan”.

A inicialização mostrada na Figura 4.10 pode ser modificada para não incluir o botão de partida e então a linha de programa de inicialização fica como descrito na Figura 4.18. Quando o primeiro ciclo do programa for executado, a variável “Passos” é falsa e o contato da negação de “Passos” é verdadeiro, logo a variável “Passo 1” torna-se verdadeiro e memoriza (Set) e a variável “Passos” também torna-se verdadeira e memoriza (Set). Quando o segundo ciclo do programa for executado, a variável “Passos” é verdadeiro e o contado da negação de “Passos” é falso, logo não executa as bobinas de saída.

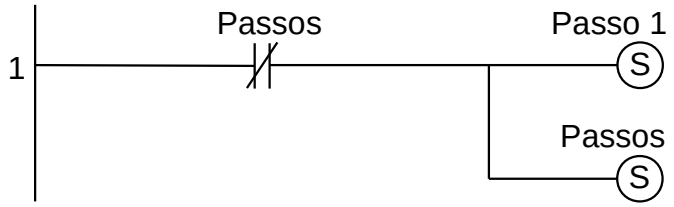

Figura 4.18 – Inicialização do programa sem contato de “partida”.

Outra solução para a inicialização, utilizando a estrutura da figura 4.10 é substituir o contato “partida” pelo contato “botão” e substituir a bobina “Passo 1” pela bobina “Passo 2”. Assim o contato “botão” é utilizado tanto para inicializar o programa quanto para sua operação.

O Passo 1 é implementado na linguagem Ladder de acordo com a Figura 4.19, onde o Passo 1 está ativo, ou seja, verdadeiro, e aguardando o operador humano pressionar o botão. Quando isso acontecer, “Passo 1” e “botão”, sendo verdadeiros, vai executar as bobinas de saída e a variável “Passo 1” torna-se falso e a variável “Passo 2” torna-se verdadeiro.

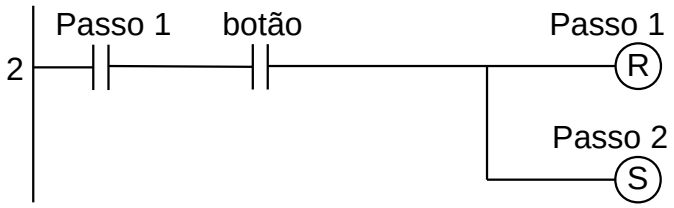

Figura 4.19 – Lógica para o Passo 1.

O Passo 2 é implementado na linguagem Ladder de acordo com a Figura 4.20, onde o Passo 2 está ativo, ou seja, verdadeiro, e aguardando o operador humano deixar de pressionar o botão. Quando isso acontecer, “Passo 2” e a negação do “botão”, sendo verdadeiros, vai executar as bobinas de saída e a variável “Passo 2” torna-se falso e a variável “Passo 3” torna-se verdadeiro.

Figura 4.20 – Lógica para o Passo 2.

O Passo 3 é implementado na linguagem Ladder de acordo com a Figura 4.21, onde o Passo 3 está ativo, ou seja, verdadeiro, e aguardando o operador humano pressionar o botão. Quando isso acontecer, “Passo 3” e “botão”, sendo verdadeiros, vai executar as bobinas de saída e a variável “Passo 3” torna-se falso e a variável “Passo 4” torna-se verdadeiro.

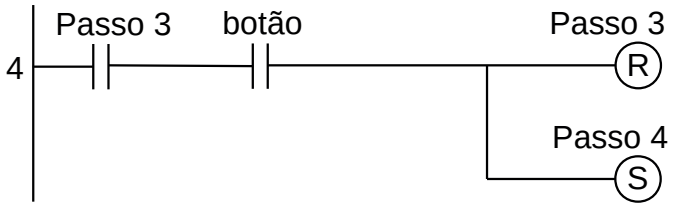

Figura 4.21 – Lógica para o Passo 3.

O Passo 4 é implementado na linguagem Ladder de acordo com a Figura 4.22, onde o Passo 4 está ativo, ou seja, verdadeiro, e aguardando o operador humano deixar de pressionar o botão. Quando isso acontecer, “Passo 4” e a negação do “botão”, sendo verdadeiros, vai executar as bobinas de saída e a variável “Passo 4” torna-se falso e a variável “Passo 1” torna-se verdadeiro.

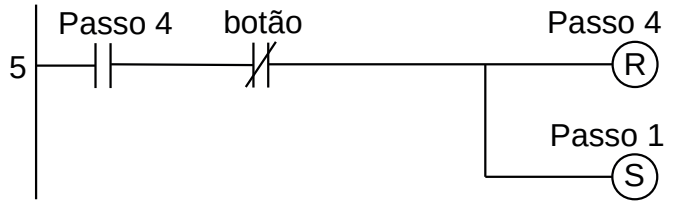

Figura 4.22 – Lógica para o Passo 4.

A saída, lâmpada, está associada aos passos que estão ativos. Isso pode ser visto na Figura 4.23.

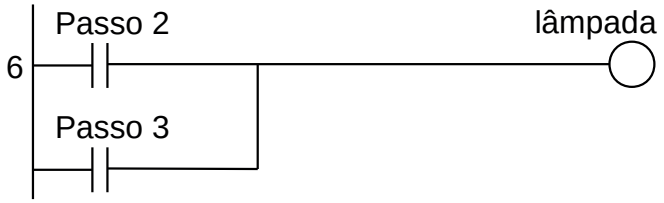

Figura 4.23 – Ativação da saída lâmpada.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 4.4 e 4.5. A tabela com as variáveis internas pode ser vista na Tabela 4.6.

Variável	Etiqueta	Descrição
M1	Passo 1	Memória para o passo 1.
M2	Passo 2	Memória para o passo 2.
M3	Passo 3	Memória para o passo 3.
M4	Passo 4	Memória para o passo 4.
M5	Passos	Memória para o programa em funcionamento.

Tabela 4.6 – Descrição das variáveis internas.

4.3.3 – Exemplo 3. Automação de um sistema de limpeza de peças

Esse exemplo mostra uma máquina mais complexa com muitos passos e com as saídas ativadas por vários passos.

Descrição operacional da máquina ou processo

O sistema de limpeza é composto por dois tanques com mistura de líquidos apropriados para limpeza. Uma talha com dois motores elétricos transporta um cesto com as peças a serem limpas desde a área de carga e descarga e imerge o cesto com as peças no tanque 1 e 2. Depois da limpeza a talha movimenta o cesto de volta para a área de carga e descarga. Os motores elétricos são acionados através de contatores identificados com as etiquetas “Xdir” para movimento para direita, “Xesq” para movimento para a esquerda, “Ysobe” para movimento de subida e “Ydesce” para movimento de descida. Existem 3 sensores de posição para parada da talha, identificados com as etiquetas “X1” para parada na área de carga e descarga, “X2” para parada sobre o Tanque 1 e “X3” para parada sobre o Tanque 2. A talha possui um sensor de comprimento do cabo de aço com saídas identificadas com as etiquetas “Ya” para quando o gancho da talha está no alto e “Yb” para quando o gancho da talha está em baixo. Existe uma botoeira identificada com a etiqueta “Limpeza” para ativar o processo de limpeza. Um diagrama esquemático representando o processo pode ser visto na Figura 4.24.

Descrição operacional da máquina ou processo

Considere como condição inicial que o gancho da talha encontra-se na área de carga e descarga, ou seja, “X1” é verdadeiro e “Yb” é verdadeiro. O fornecimento de energia elétrica é do tipo sem interrupção e a troca dos líquidos de limpeza é realizado por outro sistema.

Todo movimento do carro da talha no trilho (sentido X) deve ser feito com o gancho da talha no alto. Toda vez que o carro da talha parar, um tempo de 10 segundos deve ser feito antes do movimento de descida do gancho da talha para minimizar oscilações do cabo de aço do gancho da talha. Toda vez

que o cesto de peças for retirado de um dos tanques de limpeza, um tempo de 10 segundos deve ser feito para garantir o fim do gotejamento do produto de limpeza.

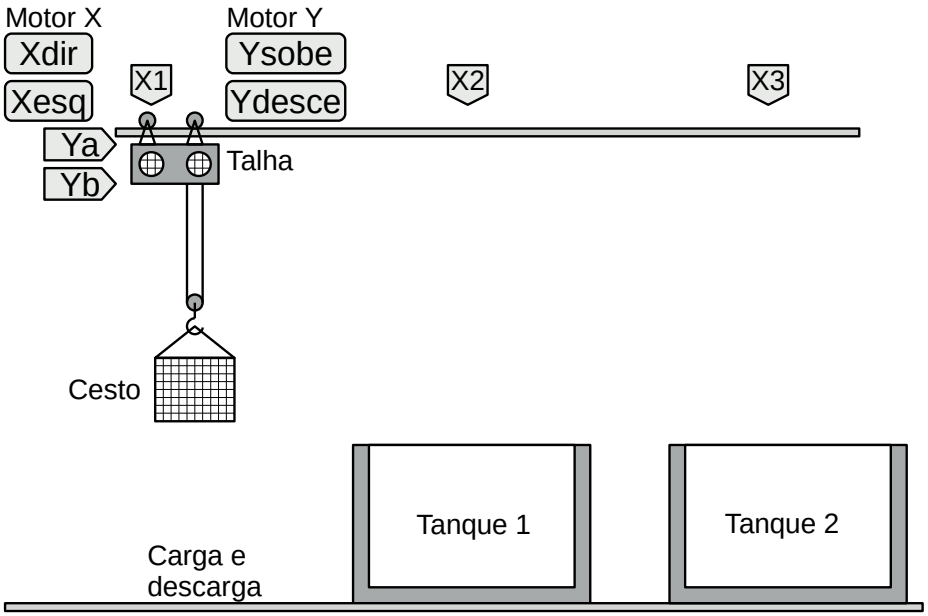

Figura 4.24 – Processo de limpeza de peças com 2 tanques.

O processo de limpeza é:

a) O cesto de peças para limpeza é colocado e retirado do gancho da talha na área de carga e descarga.

b) A primeira limpeza é no Tanque 1 e o cesto de peças deve permanecer imerso por 30 segundos.

c) A segunda limpeza é no Tanque 2 e o cesto de peças deve permanecer imerso por 60 segundos.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na Tabela 4.7.

Parafuso	Etiqueta	Descrição
I01	X1	Sensor de fim-de-curso.
I02	X2	Sensor de fim-de-curso.
I03	X3	Sensor de fim-de-curso.
I04	Ya	Sensor de comprimento de cabo.
I05	Yb	Sensor de comprimento de cabo.
I06	Limpeza	Botoeira NA

Tabela 4.7 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na Tabela 4.2.

Parafuso	Etiqueta	Descrição
Q01	Xdir	Contator de acionamento do motor.
Q02	Xesq	Contator de acionamento do motor.
Q03	Ysobe	Contator de acionamento do motor.
Q04	Ydesce	Contator de acionamento do motor.

Tabela 4.8 – Descrição das saídas.

Solução por sequência de passos

Os passos de operação da máquina podem ser identificados de acordo com cada posição e movimentação do gancho e carro da talha.

A máquina está parada

1) O operador humano coloca o cesto com as peças para limpeza no gancho da talha e aperta o botão "Limpeza".

Movimentação do cesto de peças da área de carga até o Tanque 1

2) O contator "Ysobe" é ativado e o Motor Y movimenta o gancho da talha para cima. Quando o sensor "Ya" for verdadeiro, o contator "Ysobe" é desligado e o gancho da talha para.
3) O contator "Xdir" é ativado e o Motor X movimenta o carro da talha para a direita. Quando o sensor "X2" for verdadeiro, o contator "Xdir" é desligado e o carro da talha para.
4) O carro da talha deve ficar parado por 10 segundos antes de iniciar o movimento de descida do gancho da talha para minimizar oscilações do cabo de aço devido ao movimento horizontal.
5) O contator "Ydesce" é ativado e o Motor Y movimenta o gancho da talha para baixo. Quando o sensor "Yb" for verdadeiro, o contator "Ydesce" é desligado e o gancho da talha para.

O cesto de peças está imerso no Tanque 1

6) A talha deve ficar parada por 30 segundos para garantir o processo de limpeza no Tanque 1.

Movimento do cesto de peças do Tanque 1 até o Tanque 2

7) O contator "Ysobe" é ativado e o Motor Y movimenta o gancho da talha para cima. Quando o sensor "Ya" for verdadeiro, o contator "Ysobe" é desligado e o gancho da talha para.
8) A talha deve ficar parada por 10 segundos para garantir o fim do gotejamento de produto de limpeza do Tanque 1.
9) O contator "Xdir" é ativado e o Motor X movimenta o carro da talha para a direita. Quando o sensor "X3" for verdadeiro, o contator "Xdir" é desligado e o carro da talha para.
10) O carro da talha deve ficar parado por 10 segundos antes de iniciar o movimento de descida do gancho da talha para minimizar oscilações do cabo de aço devido ao movimento horizontal.
11) O contator "Ydesce" é ativado e o Motor Y movimenta o gancho da talha para baixo. Quando o sensor "Yb" for verdadeiro, o contator "Ydesce" é desligado e o gancho da talha para.

O cesto de peças está imerso no Tanque 2

12) A talha deve ficar parada por 60 segundos para garantir o processo de limpeza no Tanque 2.

Movimento do cesto de peças do Tanque 2 até a área de descarga

13) O contator “Ysobe” é ativado e o Motor Y movimenta o gancho da talha para cima. Quando o sensor “Ya” for verdadeiro, o contator “Ysobe” é desligado e o gancho da talha para.

14) A talha deve ficar parada por 10 segundos para garantir o fim do gotejamento de produto de limpeza do Tanque 2.

15) O contator “Xesq” é ativado e o Motor X movimenta o carro da talha para a esquerda. Quando o sensor “X1” for verdadeiro, o contator “Xesq” é desligado e o carro da talha para.

16) O carro da talha deve ficar parado por 10 segundos antes de iniciar o movimento de descida do gancho da talha para minimizar oscilações do cabo de aço devido ao movimento horizontal.

17) O contator “Ydesce” é ativado e o Motor Y movimenta o gancho da talha para baixo. Quando o sensor “Yb” for verdadeiro, o contator “Ydesce” é desligado e o gancho da talha para.

Fim do processo de limpeza e o controle volta para o início.

A estrutura de passos pode ser montada usando o seguinte método:

Passo 1	Nenhuma ação. Se (“Limpeza” for verdadeiro) então vá para o Passo 2.
Passo 2	Ativar a saída “Ysobe”. Se (“Ya” for verdadeiro) então vá para o Passo 3.
Passo 3	Ativar a saída “Xdir”. Se (“X2” for verdadeiro) então vá para o Passo 4.
Passo 4	Ativar o temporizador Temp1, de 10 segundos. Se (“Temp1” for verdadeiro) então vá para o Passo 5.
Passo 5	Ativar a saída “Ydesce”. Se (“Yb” for verdadeiro) então vá para o Passo 6.
Passo 6	Ativar o temporizador Temp2, de 30 segundos. Se (“Temp2” for verdadeiro) então vá para o Passo 7.
Passo 7	Ativar a saída “Ysobe”. Se (“Ya” for verdadeiro) então vá para o Passo 8
Passo 8	Ativar o temporizador Temp1, de 10 segundos. Se (“Temp1” for verdadeiro) então vá para o Passo 9.
Passo 9	Ativar a saída “Xdir”. Se (“X3” for verdadeiro) então vá para o Passo 10.
Passo 10	Ativar o temporizador Temp1, de 10 segundos. Se (“Temp1” for verdadeiro) então vá para o Passo 11.
Passo 11	Ativar a saída “Ydesce”. Se (“Yb” for verdadeiro) então vá para o Passo 12.

Passo 12	Ativar o temporizador Temp3, de 60 segundos. Se ("Temp3" for verdadeiro) então vá para o Passo 13.
Passo 13	Ativar a saída "Ysobe". Se ("Ya" for verdadeiro) então vá para o Passo 14
Passo 14	Ativar o temporizador Temp1, de 10 segundos. Se ("Temp1" for verdadeiro) então vá para o Passo 15.
Passo 15	Ativar a saída "Xesq". Se ("X1" for verdadeiro) então vá para o Passo 16.
Passo 16	Ativar o temporizador Temp1, de 10 segundos. Se ("Temp1" for verdadeiro) então vá para o passo 17.
Passo 17	Ativar a saída "Ydesce". Se ("Yb" for verdadeiro) então vá para o Passo 1.

Codificação da sequência de passos na linguagem Ladder

O programa do controlador em linguagem Ladder é mostrado nas figuras de 8.25 até a Figura 4.49.

Inicialização. Vamos usar uma técnica de reaproveitamento de botoeiras. Quando o controlador for ligado nenhum passo estará ativo. Nos exemplos anteriores, foi necessário uma botoeira própria para partida do programa ou partida automática via f.s. (first scan). Na Figura 4.25, quando a botoeira "Limpeza" for pressionada pela primeira vez, ativará diretamente o Passo 2 e ativando a memória interna Passos permitindo que esta linha de programa fique inoperante quando a botoeira "Limpeza" for pressionada nas vezes seguintes. O diagrama em linguagem Ladder é mostrado na Figura 4.25.

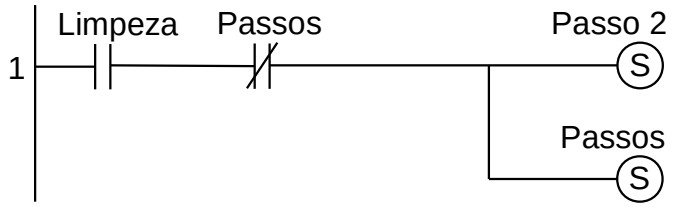

Figura 4.25 – Diagrama em linguagem Ladder para a inicialização.

Passo 1. O controlador espera uma ação do operador humano. Quando a botoeira "Limpeza" for pressionada, e portanto com lógica verdadeira, o Passo 1 é desativado e o Passo 2 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.26.

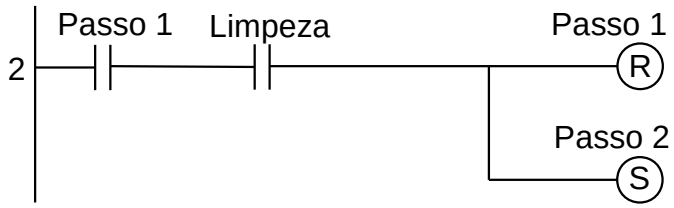

Figura 4.26 – Diagrama em linguagem Ladder para o Passo 1.

Passo 2. Movimento de subida do cesto de peças na região de carga. Veja na linha de programa 21 que a saída “Ysobe” está ativa. Quando o gancho da talha chegar no ponto alto, o sensor “Ya” é verdadeiro, o Passo 2 é desativado e o Passo 3 é ativado, a saída “Ysobe” é desativada e o movimento de subida do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.27.

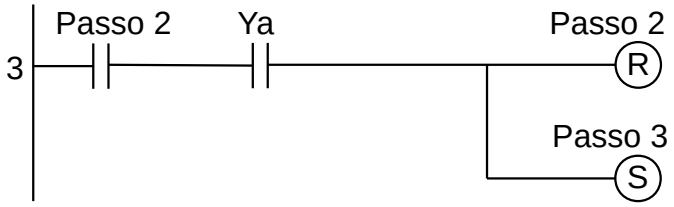

Figura 4.27 – Diagrama em linguagem Ladder para o Passo 2.

Passo 3. Movimento da talha para direita. Veja na linha de programa 19 que a saída “Xdir” está ativa. Quando o carro da talha chegar sobre o Tanque 1, o sensor “X2” é verdadeiro, o Passo 3 é desativado e o Passo 4 é ativado, a saída “Xdir”é desativada e o movimento para direita do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.28.

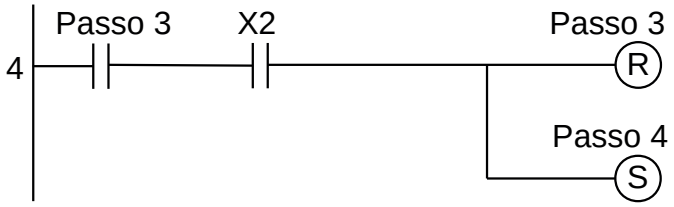

Figura 4.28 – Diagrama em linguagem Ladder para o Passo 3.

Passo 4. Parada para estabilização de movimento. Veja na linha de programa 23 que o temporizador “Temp1” está ativo. Quando o temporizador terminar a contagem, a saída “Temp1” é verdadeira, o Passo 4 é desativado e o Passo 5 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.29.

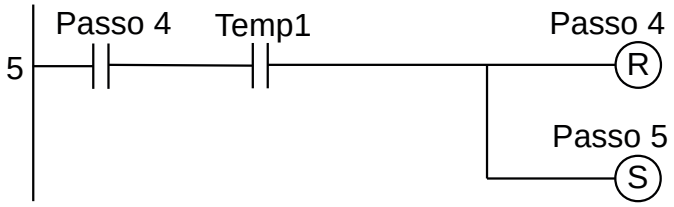

Figura 4.29 – Diagrama em linguagem Ladder para o Passo 4.

Passo 5. Movimento de descida do cesto de peças sobre o Tanque 1. Veja na linha de programa 22 que a saída “Ydesce” está ativa. Quando o gancho da talha chegar no ponto baixo, o sensor “Yb” é verdadeiro, o Passo 5 é desativado e o Passo 6 é ativado, a saída “Ydesce” é desativada e o movimento de descida do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.30.

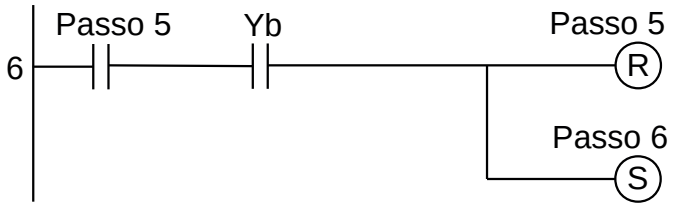

Figura 4.30 – Diagrama em linguagem Ladder para o Passo 5.

Passo 6. Banho de limpeza no Tanque 1. Veja na linha de programa 24 que o temporizador "Temp2" está ativo. Quando o temporizador terminar a contagem, a saída "Temp2" é verdadeira, o Passo 6 é desativado e o Passo 7 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.31.

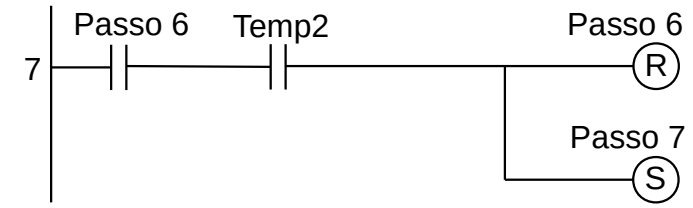

Figura 4.31 – Diagrama em linguagem Ladder para o Passo 6.

Passo 7. Movimento de subida do cesto de peças sobre o Tanque 1. Veja na linha de programa 21 que a saída "Ysobe" está ativa. Quando o gancho da talha chegar no ponto alto, o sensor "Ya" é verdadeiro, o Passo 7 é desativado e o Passo 8 é ativado, a saída "Ysobe" é desativada e o movimento de subida do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.32.

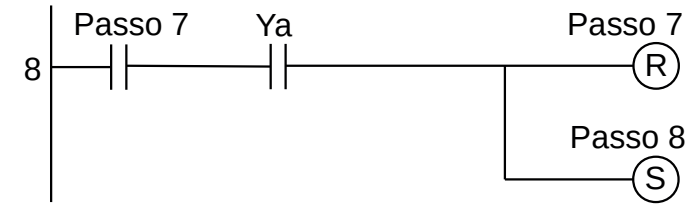

Figura 4.32 – Diagrama em linguagem Ladder para o Passo 7.

Passo 8. Parada para fim de gotejamento do produto do Tanque 1. Veja na linha de programa 23 que o temporizador "Temp1" está ativo. Quando o temporizador terminar a contagem, a saída "Temp1" é verdadeira, o Passo 8 é desativado e o Passo 9 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.33.

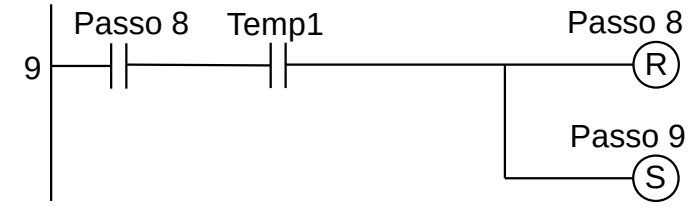

Figura 4.33 – Diagrama em linguagem Ladder para o Passo 8.

Passo 9. Movimento da talha para direita. Veja na linha de programa 19 que a saída "Xdir" está ativa. Quando o carro da talha chegar sobre o Tanque 2, o sensor "X3" é verdadeiro, o Passo 9 é desativado e o Passo 10 é ativado, a saída "Xdir"é desativada e o movimento para direita do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.34.

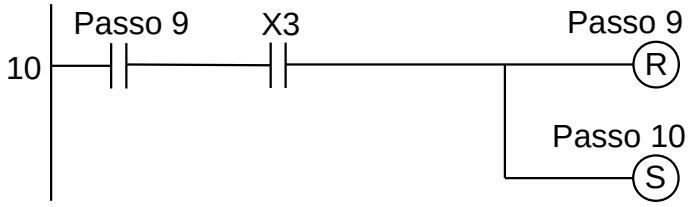

Figura 4.34 – Diagrama em linguagem Ladder para o Passo 9.

Passo 10. Parada para estabilização de movimento. Veja na linha de programa 23 que o temporizador "Temp1" está ativo. Quando o temporizador terminar a contagem, a saída "Temp1" é

verdadeira, o Passo 10 é desativado e o Passo 11 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.35.

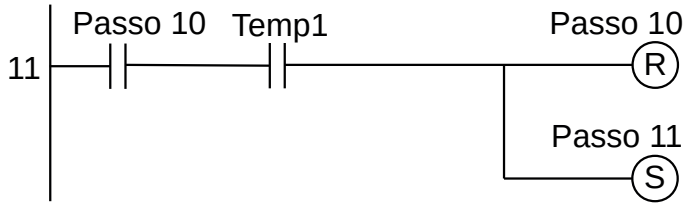

Figura 4.35 – Diagrama em linguagem Ladder para o Passo 10.

Passo 11. Movimento de descida do cesto de peças sobre o Tanque 2. Veja na linha de programa 22 que a saída "Ydesce" está ativa. Quando o gancho da talha chegar no ponto baixo, o sensor "Yb" é verdadeiro, o Passo 11 é desativado e o Passo 12 é ativado, a saída "Ydesce" é desativada e o movimento de descida do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.36.

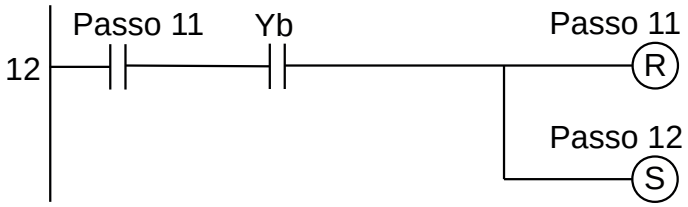

Figura 4.36 – Diagrama em linguagem Ladder para o Passo 11.

Passo 12. Banho de limpeza no Tanque 2. Veja na linha de programa 25 que o temporizador "Temp3" está ativo. Quando o temporizador terminar a contagem, a saída "Temp3" é verdadeira, o Passo 12 é desativado e o Passo 13 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.37.

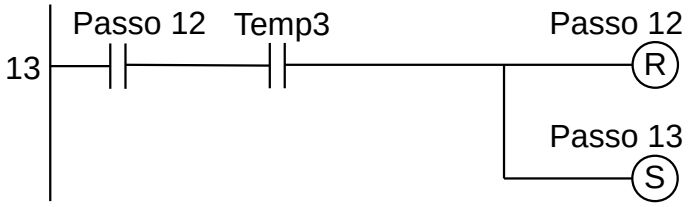

Figura 4.37 – Diagrama em linguagem Ladder para o Passo 12.

Passo 13. Movimento de subida do cesto de peças sobre o Tanque 2. Veja na linha de programa 21 que a saída "Ysobe" está ativa. Quando o gancho da talha chegar no ponto alto, o sensor "Ya" é verdadeiro, o Passo 13 é desativado e o Passo 14 é ativado, a saída "Ysobe" é desativada e o movimento de subida do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.38.

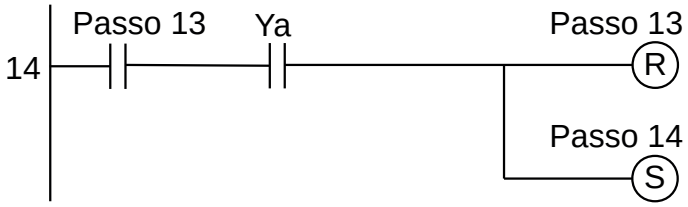

Figura 4.38 – Diagrama em linguagem Ladder para o Passo 13.

Passo 14. Parada para fim de gotejamento do produto do Tanque 2. Veja na linha de programa 23 que o temporizador "Temp1" está ativo. Quando o temporizador terminar a contagem, a saída "Temp1" é verdadeira, o Passo 14 é desativado e o Passo 15 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.39.

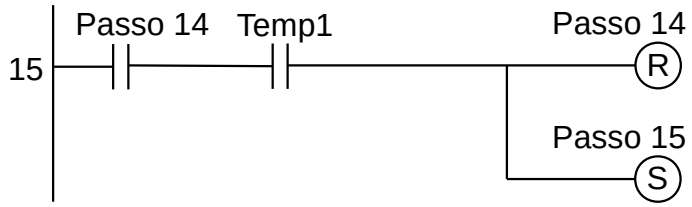

Figura 4.39 – Diagrama em linguagem Ladder para o Passo 14.

Passo 15. Movimento da talha para esquerda. Veja na linha de programa 20 que a saída "Xesq" está ativa. Quando o carro da talha chegar sobre a região de descarga, o sensor "X1" é verdadeiro, o Passo 15 é desativado e o Passo 16 é ativado, a saída "Xesq"é desativada e o movimento para esquerda do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.40.

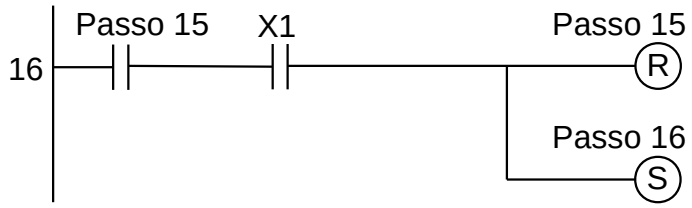

Figura 4.40 – Diagrama em linguagem Ladder para o Passo 15.

Passo 16. Parada para estabilização de movimento. Veja na linha de programa 23 que o temporizador "Temp1" está ativo. Quando o temporizador terminar a contagem, a saída "Temp1" é verdadeira, o Passo 16 é desativado e o Passo 17 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.41.

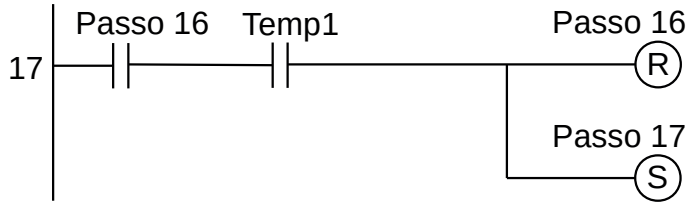

Figura 4.41 – Diagrama em linguagem Ladder para o Passo 16.

Passo 17. Movimento de descida do cesto de peças sobre a região de descarga. Veja na linha de programa 22 que a saída "Ydesce" está ativa. Quando o gancho da talha chegar no ponto baixo, o sensor "Yb" é verdadeiro, o Passo 17 é desativado e o Passo 1 é ativado, a saída "Ydesce" é desativada e o movimento de descida do cesto de peças é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.42.

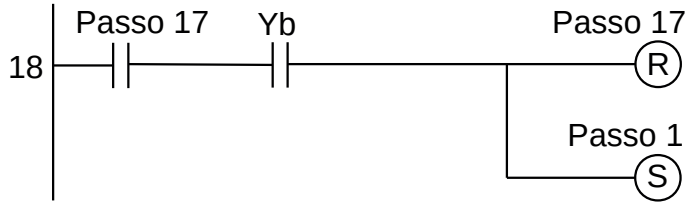

Figura 4.42 – Diagrama em linguagem Ladder para o Passo 17.

As saídas são ativadas, ou seja, tornam-se verdadeiras, de acordo com a evolução da sequência de passos.

Saída Xdir. Esta saída torna-se ativa quando o passo 3 ou o passo 9 está ativo. O diagrama em linguagem Ladder é mostrado na Figura 4.43.

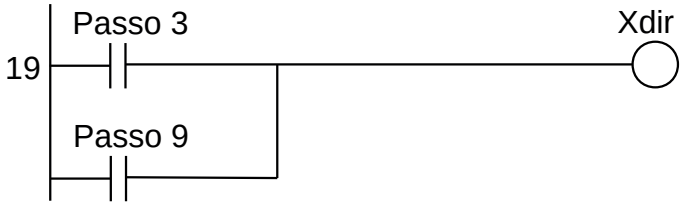

Figura 4.43 – Diagrama em linguagem Ladder para a saída Xdir.

Saída Xesq. Esta saída torna-se ativa quando o passo 15 está ativo. O diagrama em linguagem Ladder é mostrado na Figura 4.44.

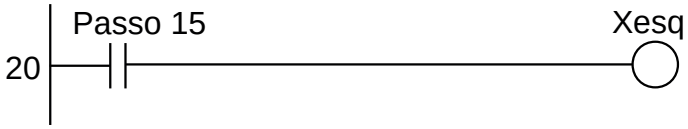

Figura 4.44 – Diagrama em linguagem Ladder para a saída Xesq.

Saída Ysobe. Esta saída torna-se ativa quando o passo 2 ou passo 7 ou passo 13 está ativo. O diagrama em linguagem Ladder é mostrado na Figura 4.45.

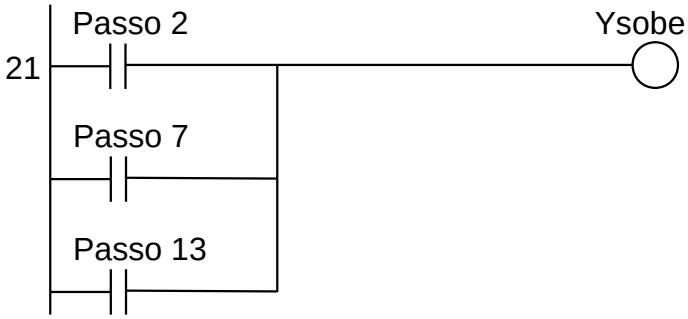

Figura 4.45 – Diagrama em linguagem Ladder para a saída Ysobe.

Saída Ydesce. Esta saída torna-se ativa quando o passo 5 ou passo 11 ou passo 17 está ativo. O diagrama em linguagem Ladder é mostrado na Figura 4.46.

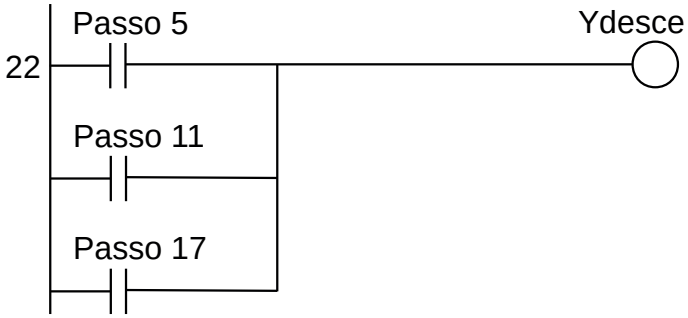

Figura 4.46 – Diagrama em linguagem Ladder para a saída Ydesce.

Temporizador Temp1 de 10 segundos. Esta saída torna-se ativa quando o passo 4 ou passo 8 ou passo 10 ou passo 14 ou passo 16 está ativo. O diagrama em linguagem Ladder é mostrado na Figura 4.47.

Observação: nesta linha de programa foi feito o reaproveitamento de um temporizador, ou seja, o mesmo temporizador de 10 segundos é utilizado em 5 momentos, ou passos, distintos. Isso somente é possível se o uso deste temporizador for feito em passos não consecutivos. Caso contrário será necessário utilizar outro temporizador. Exemplo: num "passo x" é utilizado um temporizador de 10 segundos e no "passo x+1" é utilizado um temporizador também de 10 segundos. Então no "passo x" será utilizado o temporizador "temp1" e no "passo x+1" será utilizado o temporizador "temp2".

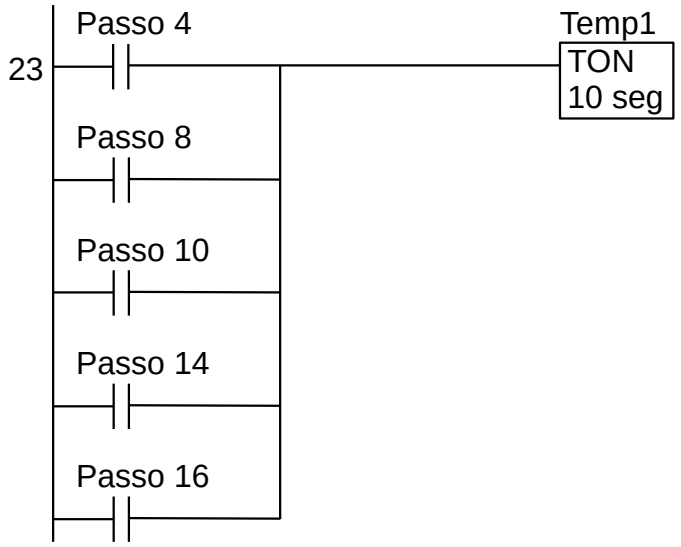

Figura 4.47 – Diagrama em linguagem Ladder para o temporizador Temp1.

Temporizador Temp2 de 30 segundos. Esta saída torna-se ativa quando o passo 6 está ativo. O diagrama em linguagem Ladder é mostrado na Figura 4.48.

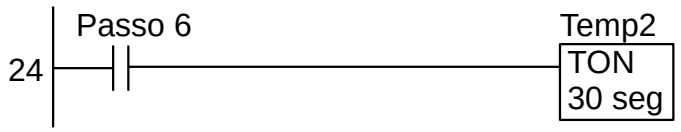

Figura 4.48 – Diagrama em linguagem Ladder para o temporizador Temp2.

Temporizador Temp3 de 60 segundos. Esta saída torna-se ativa quando o passo 12 está ativo. O diagrama em linguagem Ladder é mostrado na Figura 4.49.

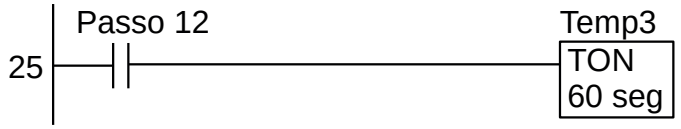

Figura 4.49 – Diagrama em linguagem Ladder para o temporizador Temp3.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 4.7 e 4.8. A tabela com as variáveis internas pode ser vista na Tabela 4.9.

Variável	Etiqueta	Descrição
M1	Passo 1	Memória para o passo 1.
M2	Passo 2	Memória para o passo 2.
M3	Passo 3	Memória para o passo 3.
M4	Passo 4	Memória para o passo 4.
M5	Passo 5	Memória para o passo 5
M6	Passo 6	Memória para o passo 6.
M7	Passo 7	Memória para o passo 7.
M8	Passo 8	Memória para o passo 8.
M9	Passo 9	Memória para o passo 9.
M10	Passo 10	Memória para o passo 10.
M11	Passo 11	Memória para o passo 11.
M12	Passo 12	Memória para o passo 12.
M13	Passo 13	Memória para o passo 13.
M14	Passo 14	Memória para o passo 14.
M15	Passo 15	Memória para o passo 15.
M16	Passo 16	Memória para o passo 16.
M17	Passo 17	Memória para o passo 17.
M18	Passos	Memória para o programa em funcionamento.
T1	Temp1	Temporizador para ligar (TON) 10 segundos.
T2	Temp2	Temporizador para ligar (TON) 30 segundos.
T3	Temp3	Temporizador para ligar (TON) 60 segundos.

Tabela 4.9 – Descrição das variáveis internas.

4.4 – Sistema complexo

Algumas máquinas ou processos possuem partes que podem ser controladas individualmente ou que possui vários atuadores funcionando ao mesmo tempo. Nestes casos existem duas possibilidades de solução: cada parte de máquina possui o próprio controlador ou um mesmo controlador controla todos os processos ao mesmo tempo.

Individualizar cada parte da máquina com seu próprio controlador tem a vantagem de utilizar controladores monolíticos de poucas saídas digitais, de 4 a 6 saídas, e com poucas entradas digitais ou analógicas, normalmente de 6 a 8 entradas. Uma das vantagens é que esses controladores são de custo menor e ocupam menos espaço no painel de controle. Outra vantagem é a paralisação de apenas uma parte da máquina em caso de defeito. Em desvantagem temos o uso de uma quantidade maior de controladores, que podem tem um ciclo de vida menor e maior dificuldade durante o processo de troca de controlador.

Então, com o uso de um controlador maior temos mais entrada e mais saídas e isso pode permitir controlar vários processos simultâneos. Uma das técnicas consiste na identificação de cada tipo de programa, sendo possível que cada programa utilize um método diferente.

Ao utilizar o método de sequência de passos, é aconselhável identificar os passos por letras e números ou outra sequência que diferencie um programa do outro programa.

4.4.1 – Exemplo 4. Automação de um estacionamento

Esse exemplo mostra a solução de um problema com várias partes do processo operando em simultâneo. É confeccionado uma modelagem e um programa específico para cada parte.

Considere o problema de automatizar um ponto de acesso a um estacionamento composto por duas barreiras e um semáforo indicativo de vagas disponíveis.

Descrição da máquina ou processo

O sistema é composto por barreiras automáticas para controle de entrada e saída de veículos. A duas barreiras são iguais, diferenciando apenas na etiqueta de identificação para os dispositivos. Assim, a barreia de saída possui um motor de indução de duplo enrolamento que permite o giro nos dois sentidos. Esse motor é ativado por dois relés, identificados como “Sab” que, quando acionado, ativa o motor a girar num determinado sentido e abrir a barreira e o outro relé identificado com “Sfe” que, quando acionado, ativa o motor a girar num determinado sentido e fechar a barreira. Existem dois sensores que são acionados quando a barreira está fechada “Sf” e quando a barreira está aberta “Sa”. Para detectar o veículo que deseja sair, existe um sensor identificado com a etiqueta “S0” que informa ao controlador que o veículo está antes da barreira. Para detectar que o veículo saiu, existe um sensor identificado com a etiqueta “S1” que informa ao controlador que o veículo está após a barreira. De maneira similar a barreira para entrada possui os mesmos sensores e atuadores, substituindo a letra S pela letra E. Existe também um semáforo composto por duas luminárias, sendo uma na cor verde identificada com a etiqueta “Sim” indicando que o estacionamento possui vagas disponíveis. Outra luminária na cor vermelha e identificada com a etiqueta “Não” indicando que o estacionamento está lotado e não possui vagas disponíveis. Um diagrama esquemático pode ser visto na Figura 4.50.

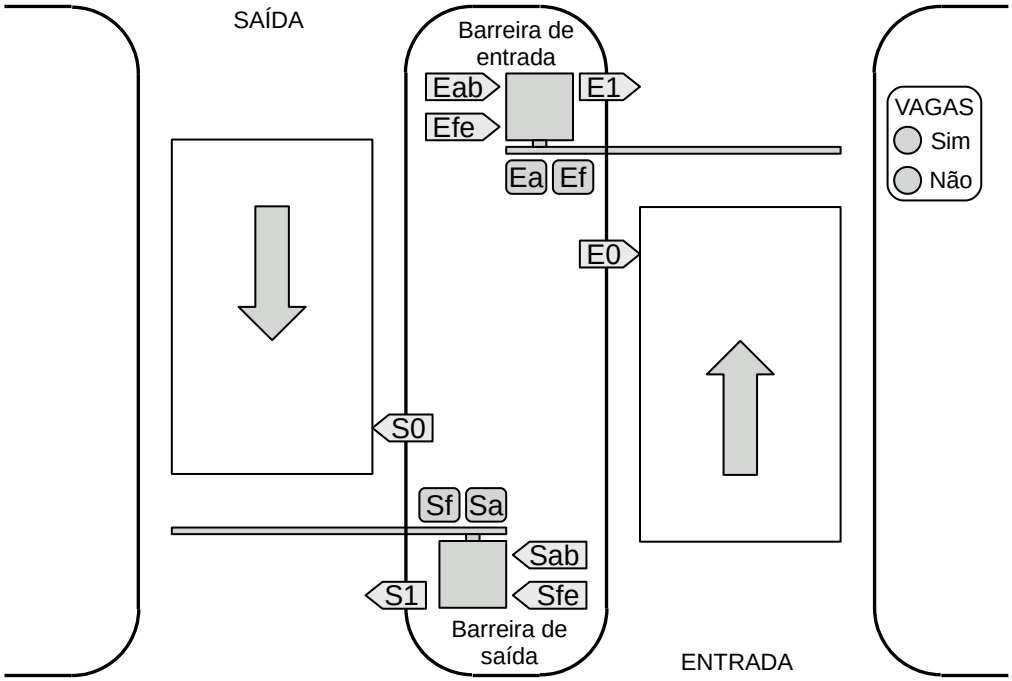

Figura 4.50 – Diagrama esquemático de um ponto de acesso para um estacionamento.

Descrição operacional da máquina ou processo

Os dispositivos do ponto de acesso ao estacionamento podem ser divididos em três partes: barreira de saída, barreira de entrada e semáforo de vagas.

O funcionamento da barreira de saída é descrito por:

a) O veículo entra na região retangular de frente da barreira de saída. O veículo é detectado pelo sensor que está antes da barreira.

b) O controlador abre a barreira e fica esperando o veículo sair.

c) O veículo sai do estacionamento. O veículo é detectado pelo sensor que está depois da barreira.

O funcionamento da barreira de entrada é descrito por:

a) O veículo entra na região retangular de frente da barreira de entrada. O veículo é detectado pelo sensor que está antes da barreira.

b) Se existir vaga no estacionamento, o controlador abre a barreira e fica esperando o veículo entrar.

c) O veículo entra no estacionamento. O veículo é detectado pelo sensor que está depois da barreira.

O funcionamento do semáforo de vagas é descrito por:

a) Quando o controlador é inicializado, a quantidade de veículos estacionados é zerado ou a quantidade de vagas é máximo. A lâmpada sinalizadora de vagas "Sim" é ativado.

b) Quando o veículo entra no estacionamento, um sinal de adição de veículo ou sinal de redução de vagas é enviado a um contador.

c) Quando o veículo sai do estacionamento, um sinal de redução de veículo ou sinal de adição de vagas é enviado a um contador.

d) Se a quantidade de veículos for igual à quantidade de vagas ou a quantidade de vagas for zero, então é feito a troca de ativação das lâmpadas sinalizadoras.

Observação importante: este projeto não contempla possíveis erros de movimentação do motorista do veículo. Considere que ações adversas não vão ocorrer, como por exemplo, o motorista fazer movimento de marcha a ré quando tocar os sensores de antes e depois da barreira, ou ficar parado por muito tempo sobre os sensores de antes e depois da barreira. Também não existem condições para análise de defeitos na movimentação da barreira.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na Tabela 4.10.

Parafuso	Etiqueta	Descrição
I01	E0	Sensor magnético de presença de veículos.
I02	E1	Sensor magnético de presença de veículos.
I03	Ea	Sensor de fim-de-curso.
I04	Ef	Sensor de fim-de-curso.
I05	S0	Sensor magnético de presença de veículos.
I06	S1	Sensor magnético de presença de veículos.
I07	Sa	Sensor de fim-de-curso.
I08	Sf	Sensor de fim-de-curso.
I09	partida	Boteira NA.

Tabela 4.10 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na Tabela 4.11.

Parafuso	Etiqueta	Descrição
Q01	Eab	Contator de acionamento do motor.
Q02	Efe	Contator de acionamento do motor.
Q03	Sab	Contator de acionamento do motor.
Q04	Sfe	Contator de acionamento do motor.
Q05	Sim	Lâmpada sinalizadora.
Q06	Não	Lâmpada sinalizadora.

Tabela 4.11 – Descrição das saídas.

Solução por sequência de passos

Da observação do funcionamento da máquina é possível identificar 3 blocos de atividades: barreira de entrada, barreira de saída, semáforo.

A barreira de saída pode ser representada pela sequência de passos:

Passo S1. O controlador da barreira fica esperando até que um veículo se aproxime para sair e o sensor “S0” seja verdadeiro.

Passo S2. O motor da barreira é acionado através do “Sab” para abrir a barreira até que a barreira fique totalmente aberta e o sensor “Sa” seja verdadeiro.

Passo S3. O controlador da barreira fica esperando até que o veículo começa a sair e o sensor “S1” seja verdadeiro.

Observe que o fato do sensor “S1” ser verdadeiro não significa que o veículo saiu totalmente, significa que o veículo começou a sair. O veículo terá saído totalmente quando “S1” tornar-se falso.

Passo S4. O controlador da barreira fica esperando até que o veículo saia totalmente do estacionamento e o sensor “S1” for falso.

Passo S5. O motor da barreira é acionado através do “Sfe” para fechar a barreira até que a barreira fique totalmente fechada e o sensor “Sf” seja verdadeiro.

A barreira de entrada pode ser representada pela sequência de passos:

Passo E1. O controlador da barreira fica esperando até que um veículo se aproxime para entrar, o sensor “E0” seja verdadeiro e o indicador de vagas esteja com o “Sim” em verdadeiro.

Passo E2. O motor da barreira é acionado através do “Eab” para abrir a barreira até que a barreira fique totalmente aberta e o sensor “Ea” seja verdadeiro.

Passo E3. O controlador da barreira fica esperando até que o veículo começa a entrar e o sensor “E1” seja verdadeiro.

Observe que o fato do sensor “E1” ser verdadeiro não significa que o veículo entrou totalmente, significa que o veículo começou a entrar. O veículo terá entrado totalmente quando “E1” tornar-se falso.

Passo E4. O controlador da barreira fica esperando até que o veículo entre totalmente no estacionamento e o sensor “E1” for falso.

Passo E5. O motor da barreira é acionado através do “Efe” para fechar a barreira até que a barreira fique totalmente fechada e o sensor “Ef” seja verdadeiro.

O semáforo é descrito através de frases lógicas:

Se início do processo **então** armazenar a quantidade de vagas no registrador do contador "Cont1".

Se um veículo entrar no estacionamento (sensor "E1" passar de verdadeiro para falso) **então** decrementar o contador "Cont1".

Se um veículo sair do estacionamento (sensor "S1" passar de verdadeiro para falso) **então** incrementar o contador "Cont1".

Se o contador de carros for menor que a quantidade de vagas (a saída de "Cont1" é falso) **então** fazer a saída "Sim" verdadeiro e a saída "Não" falso.

Se o contador de carros for igual à quantidade de vagas (a saída de "Cont1" é verdadeiro) **então** fazer a saída "Sim" falso e a saída "Não" verdadeiro.

A estrutura de passos pode ser montada usando o seguinte método:

Para a barreira de saída:

Passo S1	Nenhuma ação. Se ("S0" for verdadeiro) então vá para o passo S2.
Passo S2	Fazer "Sab" verdadeiro. Se ("Sa" for verdadeiro) então vá para o passo S3.
Passo S3	Nenhuma ação. Se ("S1" for verdadeiro) então vá para o passo S4.
Passo S4	Nenhuma ação. Se ("S1" for falso) então vá para o passo S5.
Passo S5	Fazer "Sfe" verdadeiro. Se ("Sf" for verdadeiro) então vá para o passo S1.

Para a barreira de entrada:

Passo E1	Nenhuma ação. Se ("E0" for verdadeiro e "Sim" for verdadeiro) então vá para o passo E2.
Passo E2	Fazer "Eab" verdadeiro. Se ("Ea" for verdadeiro) então vá para o passo E3.
Passo E3	Nenhuma ação. Se ("E1" for verdadeiro) então vá para o passo E4.
Passo E4	Nenhuma ação. Se ("E1" for falso) então vá para o passo E5.
Passo E5	Fazer "Efe" verdadeiro. Se ("Ef" for verdadeiro) então vá para o passo E1.

Codificação da sequência de passos na linguagem Ladder

Para implementar o programa controlador para o semáforo é necessário conhecer as diversas implementações que este bloco pode ter em cada controlador.

De maneira genérica o bloco do contador é definido como contador progressivo ("CTU" do inglês "up counter") e contador regressivo ("CTD" do inglês "down counter"), que atuam sobre um registrador interno, que pode ser o mesmo registrador para os dois tipos de contadores. Esses contadores, de forma minimalista, possuem duas entradas, ativação e contagem, e uma saída. É necessário ativar o contador para iniciar o processo de contagem. Se esse sinal for desligado, ou falso, o contador para de contar e o valor acumulado é zerado. Também é necessário carregar um valor de contagem em um registrador interno. Esse valor pode ser fixo ou proveniente de um outro registrador externo. Alguns fabricantes implementam esses contadores em blocos separados, e outros fabricantes num mesmo bloco.

Uma simbologia para o contador progressivo pode visto na Figura 4.51, onde "CTU" é o código do tipo de contador, "Cont1" é a etiqueta da saída do contador, "nn" é o valor inteiro armazenado no registrador interno "R1". Uma simbologia para o contador regressivo pode ser visto na Figura 4.52, onde "CTD" é o código do tipo de contador, "Cont2" é a etiqueta da saída do contador, "nx" é o valor inteiro armazenado no registrador interno "R2". Para os dois tipos, o contato "Ativação" permite ativar a contagem do contador e o contato "Contagem", quando passa de falso para verdadeiro, conta uma unidade no tipo de contador.

A norma IEC-61131-3 define os formatos dos blocos de contadores, conforme mostrado no capítulo inicial, mas os fabricantes são livres para definir seu formato e variações. Alguns fabricantes aderem fielmente à norma. Outros fabricantes preferem apresentar uma gama maior de tipos de contadores.

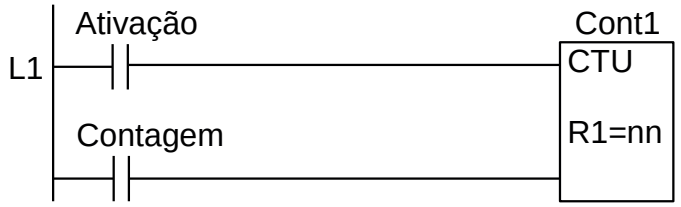

Figura 4.51 – Simbologia genérica para contador progressivo (CTU).

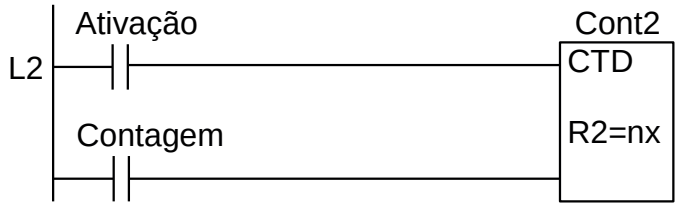

Figura 4.52 – Simbologia genérica para contador regressivo (CTD).

Quando implementados num mesmo bloco, o contador pode ter contagem progressiva e regressiva simultânea no mesmo registrador. Uma simbologia para o contador progressivo e regressivo pode visto na Figura 4.53, onde "CTUD" é o código do tipo de contador, "Cont3" é a etiqueta da saída do contador, "xx" é o valor inteiro armazenado no registrador interno "R3". O contato "Direção" é utilizado para determinar se o contador é do tipo progressivo (falso) ou regressivo (verdadeiro).

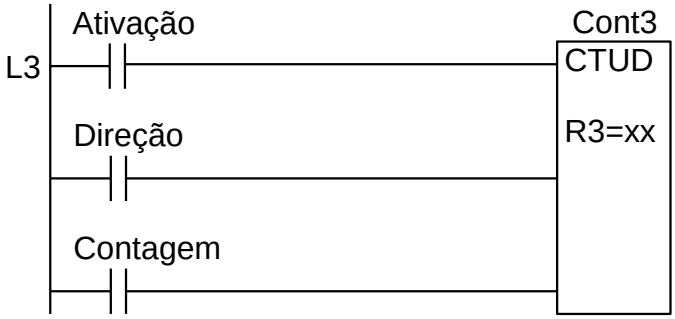

Figura 4.53 – Simbologia genérica para contador progressivo e regressivo (CTUD).

O programa do controlador em linguagem Ladder é mostrado nas figuras de 4.54 até a Figura 4.67.

Inicialização do programa. A inicialização é feita de maneira semelhante aos outros programas anteriores, mas observe que agora é dado partida em dois passos: "Passo E1" do programa de controle da barreira de entrada e "Passo S1" do programa de controle da barreira de saída. O diagrama em linguagem Ladder é mostrado na Figura 4.54.

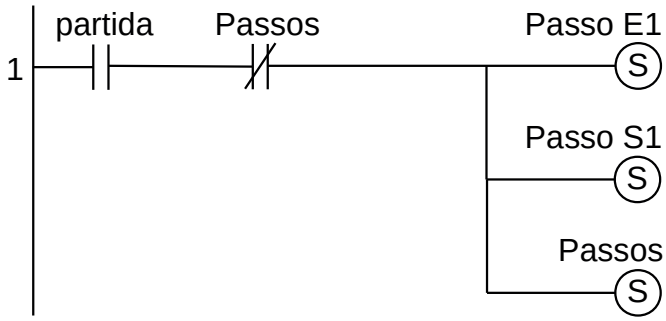

Figura 4.54 – Diagrama em linguagem Ladder para a inicialização.

Programa de controle da barreira de saída

Passo S1. O controlador espera um veículo para sair do estacionamento. Quando o sensor de veículos "S0" for ativado, e portanto com lógica verdadeira, o Passo S1 é desativado e o Passo S2 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.55.

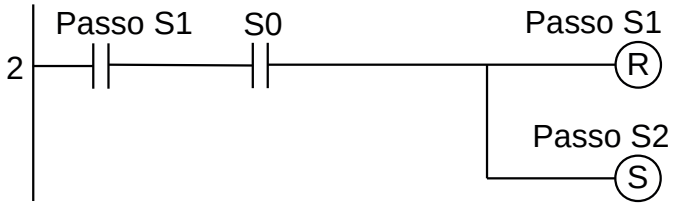

Figura 4.55 – Diagrama em linguagem Ladder para o Passo S1.

Passo S2. Movimento de abertura da barreira de saída. Veja na linha de programa 15 que a saída "Sab" está ativa. Quando a barra da barreira chegar no ponto alto, o sensor "Sa" é verdadeiro, o Passo S2 é desativado e o Passo S3 é ativado, a saída "Sab" é desativada e o movimento de abertura da barreira de saída é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.56.

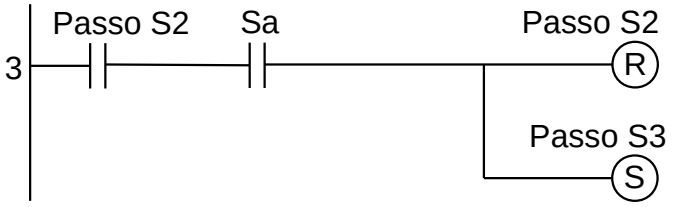

Figura 4.56 – Diagrama em linguagem Ladder para o Passo S2.

Passo S3. Início do movimento de saída do veículo. Quando o veículo começar a sair do estacionamento, o sensor "S1" é verdadeiro, o Passo S3 é desativado e o Passo S4 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.57.

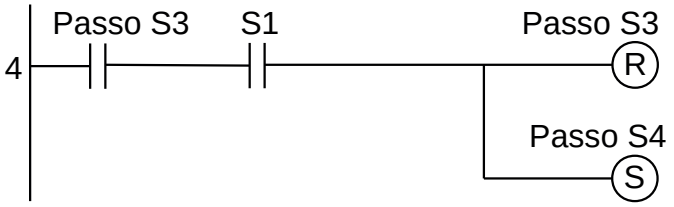

Figura 4.57 – Diagrama em linguagem Ladder para o Passo S3.

Passo S4. Final do movimento de saída do veículo. Quando o veículo sair totalmente do estacionamento, o sensor "S1" é falso, o Passo S4 é desativado e o Passo S5 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.58.

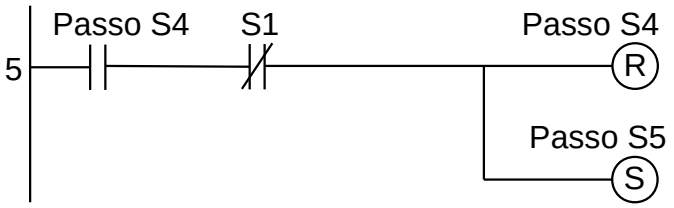

Figura 4.58 – Diagrama em linguagem Ladder para o Passo S4.

Passo S5. Movimento de fechamento da barreira de saída. Veja na linha de programa 16 que a saída "Sfe" está ativa. Quando a barra da barreira chegar no ponto baixo, o sensor "Sf" é verdadeiro, o Passo S5 é desativado e o Passo S1 é ativado, a saída "Sfe" é desativada e o movimento de fechamento da barreira de saída é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.58.

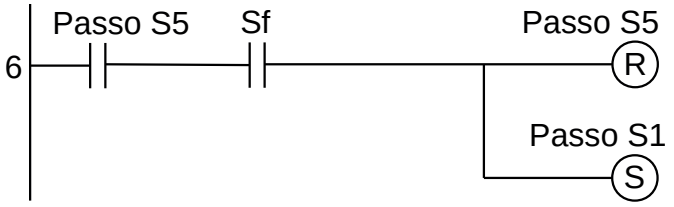

Figura 4.59 – Diagrama em linguagem Ladder para o Passo S5.

Programa de controle da barreira de entrada

Passo E1. O controlador espera um veículo para entrar no estacionamento. Quando o sensor de veículos "E0" for ativado, e portanto com lógica verdadeira, e possuir vagas no estacionamento, a saída externa "Sim" é verdadeiro, o Passo E1 é desativado e o Passo E2 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.55.

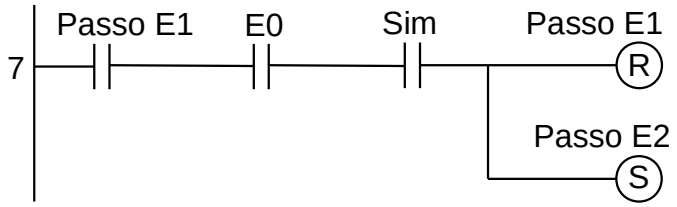

Figura 4.60 – Diagrama em linguagem Ladder para o Passo E1.

Passo E2. Movimento de abertura da barreira de entrada. Veja na linha de programa 17 que a saída "Eab" está ativa. Quando a barra da barreira chegar no ponto alto, o sensor "Ea" é verdadeiro, o Passo E2 é desativado e o Passo E3 é ativado, a saída "Eab" é desativada e o movimento de abertura da barreira de entrada é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.61.

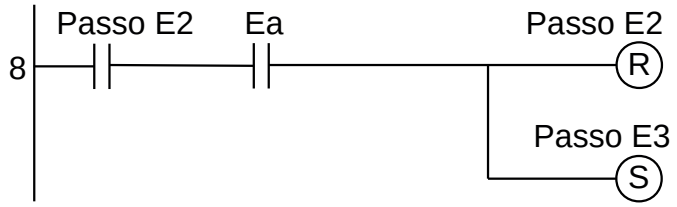

Figura 4.61 – Diagrama em linguagem Ladder para o Passo E2.

Passo E3. Início do movimento de entrada do veículo. Quando o veículo começar a entrar no estacionamento, o sensor "E1" é verdadeiro, o Passo E3 é desativado e o Passo E4 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.62.

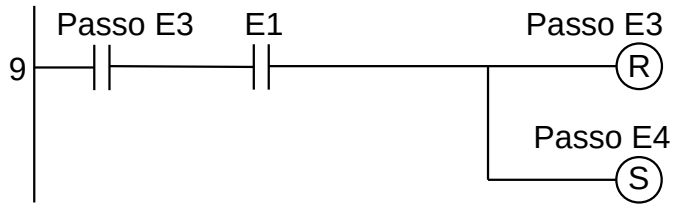

Figura 4.62 – Diagrama em linguagem Ladder para o Passo E3.

Passo E4. Final do movimento de entrada do veículo. Quando o veículo entrar totalmente no estacionamento, o sensor "E1" é falso, o Passo E4 é desativado e o Passo E5 é ativado. O diagrama em linguagem Ladder é mostrado na Figura 4.63.

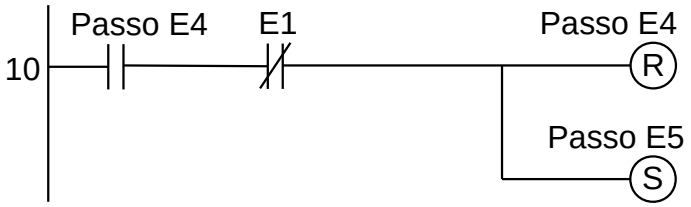

Figura 4.63 – Diagrama em linguagem Ladder para o Passo E4.

Passo E5. Movimento de fechamento da barreira de entrada. Veja na linha de programa 18 que a saída "Efe" está ativa. Quando a barra da barreira chegar no ponto baixo, o sensor "Ef" é verdadeiro, o Passo E5 é desativado e o Passo E1 é ativado, a saída "Efe" é desativada e o movimento de fechamento da barreira de entrada é interrompido. O diagrama em linguagem Ladder é mostrado na Figura 4.64.

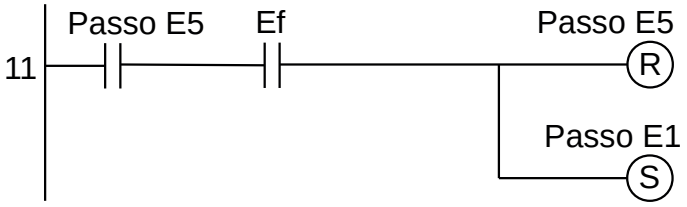

Figura 4.64 – Diagrama em linguagem Ladder para o Passo E5.

Programa de controle do semáforo

Foi adotado, aqui, o método do contador genérico progressivo e regressivo com controle de direção e contagem. Se a direção for falsa a contagem é progressiva, se a direção for verdadeira a contagem é regressiva. O Passo S5 garante que o veículo saiu totalmente do estacionamento e portanto deve ser feita uma contagem regressiva na quantidade de veículos. O Passo S6 garante que o veículo entrou totalmente no estacionamento e portanto deve ser feita uma contagem progressiva na quantidade de veículos. O diagrama em linguagem Ladder é mostrado na Figura 4.65.

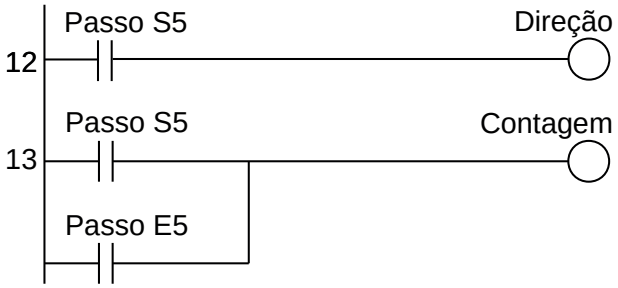

Figura 4.65 – Diagrama em linguagem Ladder para Direção e Contagem.

A contagem de veículos é realizada com o uso de um contador progressivo e regressivo com direção e contagem etiquetado como “Cont1”. Um valor inteiro é armazenado num registrador interno “R1” que é a quantidade de vagas disponíveis no estacionamento. No nosso caso foi proposto a quantidade experimental de 30 vagas. O contador permanece sempre ativo e portanto a entrada de ativação é ligada em sempre verdadeiro. O diagrama em linguagem Ladder é mostrado na Figura 4.66.

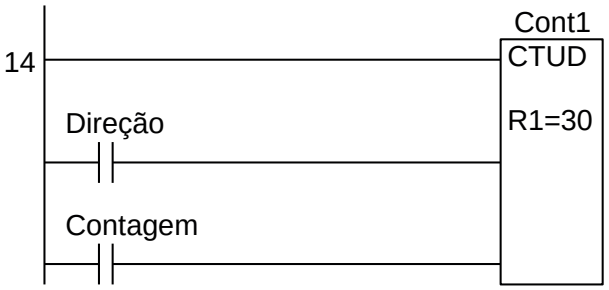

Figura 4.66 – Diagrama em linguagem Ladder para o contador progressivo e regressivo.

Ativação das saídas

As saídas são ativadas, ou seja, tornam-se verdadeiras, de acordo com a evolução da sequência de passos, exceto as saídas “Sim” e “Não” que dependem da situação do contador. Se a quantidade de veículos é menor que a quantidade de vagas, “Cont1” é falso e portanto a saída “Sim” é verdadeiro e a saída “Não” é falso. Se a quantidade de veículos é igual à quantidade de vagas, “Cont1” é verdadeiro e portanto a saída “Sim” é falsa e a saída “Não” é verdadeiro. O diagrama em linguagem Ladder é mostrado na Figura 4.67.

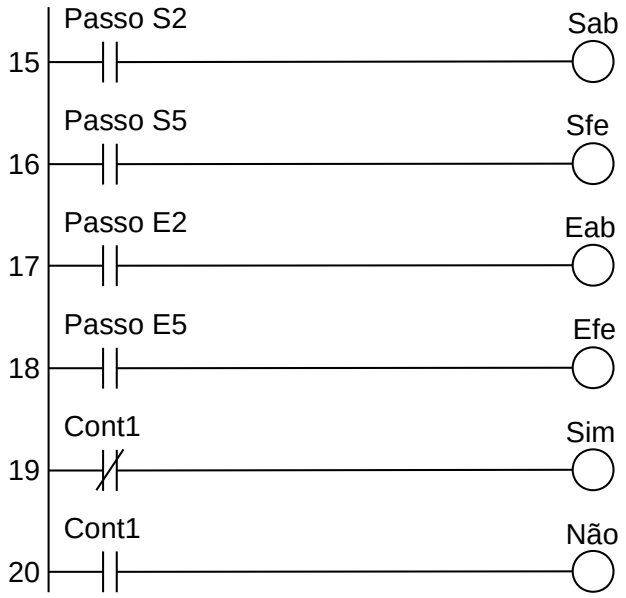

Figura 4.67 – Diagrama em linguagem Ladder para as saídas.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 8.10 e 8.11. A tabela com as variáveis internas pode ser vista na Tabela 4.12.

Variável	Etiqueta	Descrição
M1	Passo S1	Memória para o passo S1.
M2	Passo S2	Memória para o passo S2.
M3	Passo S3	Memória para o passo S3.
M4	Passo S4	Memória para o passo S4.
M5	Passo S5	Memória para o passo S5
M6	Passo E1	Memória para o passo E1.
M7	Passo E2	Memória para o passo E2.
M8	Passo E3	Memória para o passo E3.
M9	Passo E4	Memória para o passo E4.
M10	Passo E5	Memória para o passo E5.
M11	Passos	Memória para o programa em funcionamento.
M12	Direção	Memória para a direção de contagem.
M13	Contagem	Memória para contagem de veículos.
C1	Cont1	Contador progressivo e regressivo.

Tabela 4.12 – Descrição das variáveis internas.

4.4.2 – Exemplo 5. Automação de um sistema de tanques de reação química

Esse exemplo mostra outro processo dividido em partes e uma das partes fica condicionado ao término das outras duas partes.

Descrição da máquina ou processo

Um sistema de tanques para reação química é composto por três sistemas, conforme mostrado na figura 4.68. Cada tanque possui um motor de indução com um agitador, sensores de nível e temperatura e a entrada e saída de produtos é realizada por bombas centrífugas acionadas por motores de indução. O tanque 1 é para uma reação química exotérmica do produto 1 com o produto 2. O tanque 2 é para aquecimento do produto 3. O tanque 3 é para uma reação química do resultado da reação do produto 1 com o produto 2 com o produto 3 aquecido.

O tanque 1 possui um contator de acionamento de motor do agitador com etiqueta “M1”, um contator de acionamento de motor de bomba centrífuga com etiqueta “B1”, um contator de acionamento de motor de bomba centrífuga com etiqueta “B2”, um sensor tipo chave de nível com etiqueta “L1b” (tanque 1 baixo), um sensor tipo chave de nível com etiqueta “L1a” (tanque 1 alto) e um sensor tipo chave de nível com etiqueta “L1v” (tanque 1 vazio).

O tanque 2 possui um contator de acionamento de motor do agitador com etiqueta “M3”, um contator de acionamento de motor de bomba centrífuga com etiqueta “B3”, um contator de acionamento da resistência de aquecimento com etiqueta “Aq”, um sensor tipo chave de nível com etiqueta “L2a” (tanque 2 alto), um sensor tipo termostato com etiqueta “T2” e um sensor tipo chave de nível com etiqueta “L2v” (tanque 2 vazio).

O tanque 3possui um contator de acionamento de motor do agitador com etiqueta “M3”, um contator de acionamento de motor de bomba centrífuga com etiqueta “B4”, um contator de acionamento de motor de bomba centrífuga com etiqueta “B5”, um contator de acionamento de motor de bomba centrífuga com etiqueta “B6” e um sensor tipo chave de nível com etiqueta “L3v” (tanque 3 vazio).

Existe um painel de controle com uma chave liga/desliga do tipo rotativa para energização do controlador, atuadores e sensores. Neste painel também possui uma botoeira para início do processo com etiqueta “Partida”, uma boteira para finalização do processo com etiqueta “Parada”.

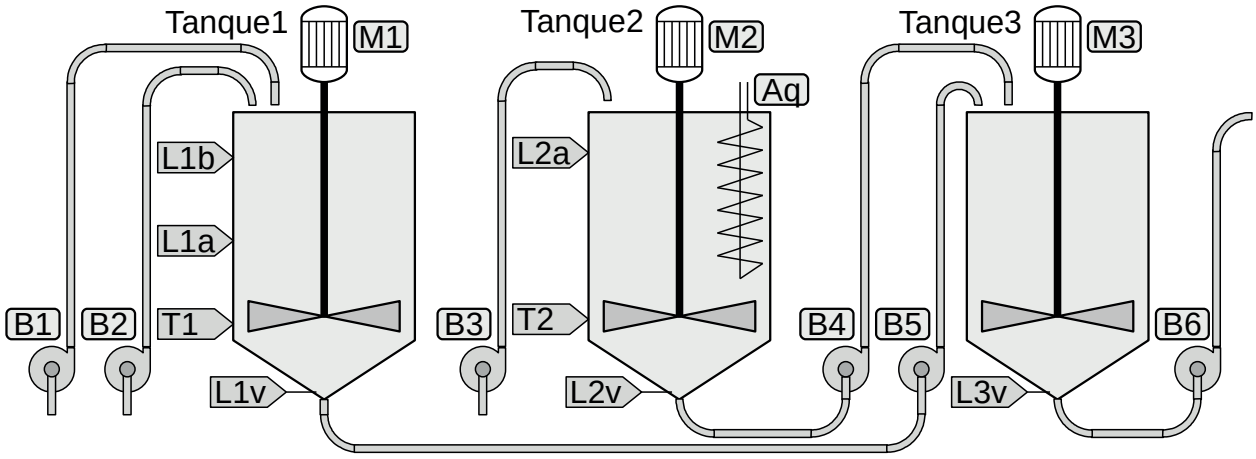

Figura 4.68 – Diagrama esquemático dos tanques de reação química.

Descrição operacional da máquina ou processo

O processo deve ser iniciado com os três tanques vazios, ou seja, “L1v” falso, “L2v” falso e “L3v” falso. Os processos químicos nos três tanques podem ocorrer em paralelo, ou seja, os tanques 1, 2 e 3 podem funcionar simultaneamente.

Operação do tanque 1:

a) A bomba B1 é ligada até o sensor de nível L1a ser verdadeiro.

b) O motor M1 do agitador é ligado.
c) A bomba B2 é ligada até o sensor de nível L1b ser verdadeiro.
d) O motor M1 do agitador é desligado quando o sensor de temperatura T1 for verdadeiro.
e) Aguardar até o sensor de nível L1v ser falso.
f) Reiniciar o processo se a finalização não foi solicitada.

Operação do tanque 2:

a) A bomba B3 é ligada até o sensor de nível L2a ser verdadeiro.
b) O motor M2 do agitador e a resistência de aquecimento Aq são ligados.
c) O motor M2 do agitador e a resistência de aquecimento Aq são desligados quando o sensor de temperatura T2 for verdadeiro.
d) Aguardar até o sensor de nível L2v ser falso.
e) Reiniciar o processo se a finalização não foi solicitada.

Operação do tanque 3:

a) Aguardar o término do processo do tanque 1 (etapa "e") e do tanque 2 (etapa "d").
b) A bomba B4 é ligada até o sensor de nível L1v ser falso.
c) O motor M3 do agitador é ligado.
d) A bomba B5 é ligada até o sensor de nível L2v ser falso.
e) O motor M3 do agitador é desligado quando o tempo de agitação de 5 minutos terminar.
f) A bomba B6 é ligada até o sensor de nível L3v ser falso.
g) Reiniciar o processo se a finalização não foi solicitada.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 4.13.

Parafuso	Etiqueta	Descrição
I01	L1a	Sensor chave de nível.
I02	L1b	Sensor chave de nível.
I03	L2a	Sensor chave de nível.
I04	L1v	Sensor chave de nível.
I05	L2v	Sensor chave de nível.
I06	L3v	Sensor chave de nível.
I07	T1	Sensor termostato.
I08	T2	Sensor termostato.
I09	Partida	Boteira N.A.
I10	Parada	Boteira N.A.

Tabela 4.13 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 4.14.

Parafuso	Etiqueta	Descrição
Q01	M1	Contator para motor de indução.
Q02	M2	Contator para motor de indução.
Q03	M3	Contator para motor de indução.
Q04	B1	Contator para motor de indução.
Q05	B2	Contator para motor de indução.
Q06	B3	Contator para motor de indução.
Q07	B4	Contator para motor de indução.
Q08	B5	Contator para motor de indução.
Q09	B6	Contator para motor de indução.
Q10	Aq	Contator para resistência aquecimento.

Tabela 4.14 – Descrição das saídas.

A solução deste problema é realizada utilizando os métodos de frases lógicas e sequência de passos. A codificação em linguagem Ladder é feita imediatamente após a montagem da frase lógica e do passo da sequência de passos.

Inicialização

Realizado através de frases lógicas.

Após energizar o controlador, o operador pressiona a botoeira "Partida". Esse evento deve ser do tipo "primeiro ciclo" e a botoeira fica desativada nos ciclos seguintes através de uma memória de trava. São feitos verdadeiros os passos iniciais dos programas sequenciais. Também é feito verdadeiro a memória de trava (bobina interna) com a etiqueta "Trava1". Observe que, devido às exigências do projeto, foi criada a variável "Hab" para habilitar o início do funcionamento do programa apenas quando os três tanques estiverem vazios.

1) Se o tanque 1 estiver vazio e o tanque 2 estiver vazio e o tanque 3 estiver vazio então habilitar o funcionamento do programa.

SE ("L1v" é falso e "L2v" é falso e "L3v" é falso) **ENTÃO** (fazer "Hab" verdadeiro).

2) Se a botoeira de partida for pressionada pela primeira vez e o funcionamento do programa estiver habilitado então iniciar as sequências de passos.

SE ("Partida" é verdadeiro e "Hab" é verdadeiro e "Trava1" é falso) **ENTÃO** (fazer Passo11 verdadeiro e memorizar, fazer Passo21 verdadeiro e memorizar, fazer o Passo31 verdadeiro e memorizar e fazer Trava1 verdadeiro e memorizar).

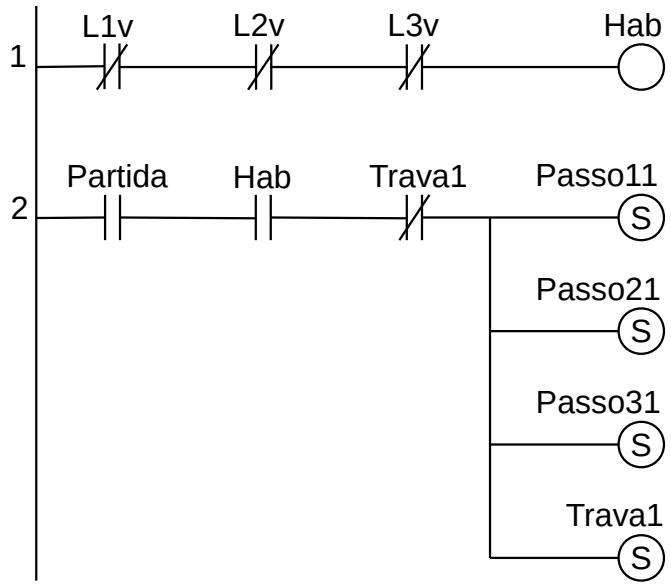

Figura 4.69 – Programa em linguagem Ladder para a frase lógica 1 e 2.

A botoeira de "Partida" também torna verdadeira a memória de funcionamento com etiqueta "Func". Essa memória será utilizada pelos programas sequenciais para reiniciar o processo.

Quando a botoeira "Parada" for pressionada a memória "Func" é feita falsa e isso impede que os programas sequenciais sejam reiniciados. A botoeira "Partida" pode ser utilizada novamente quando todos os programas sequenciais estiverem terminados, ou seja, quando todos os três tanques estiverem vazios, habilitando o reinício dos programas através da variável "Hab".

3) Se a botoeira de partida for pressionada e o funcionamento do programa estiver habilitado então uma variável indicadora de funcionamento é feita verdadeira.

SE ("Partida" é verdadeiro e "Hab" é verdadeiro) **ENTÃO** (fazer "Func" verdadeiro e memorizar).

4) Se a botoeira de parada for pressionada então variável indicadora de funcionamento é feita falsa.

SE ("Parada" é verdadeiro) **ENTÃO** (fazer "Func" falso).

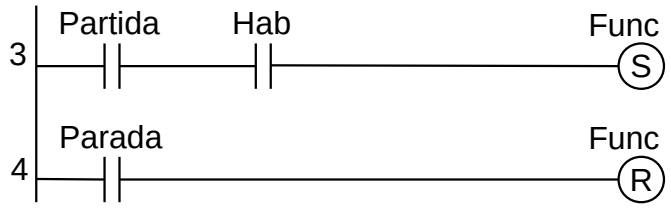

Figura 4.70 – Programa em linguagem Ladder para a frase lógica 3 e 4.

Observe que a variável "Func" pode ser analisada no início ou final da sequência de passos. Na solução deste problema optou-se por fazer a análise no final de cada sequência.

Automação do tanque 1

1) "A bomba B1 é ligada até o nível L1a ser verdadeiro."

Passo 11	Ativar o contator da bomba B1.
	Se ("L1a" for verdadeiro) então vá para o Passo 12.

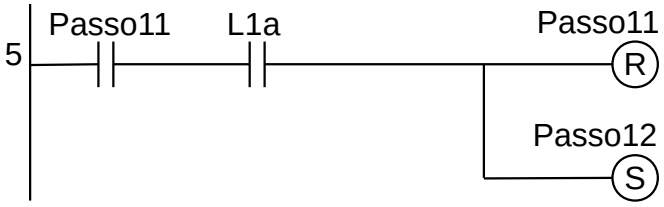

Figura 4.71 – Programa em linguagem Ladder para ao Passo 11.

2) "O motor M1 do agitador é ligado." É necessário a memorização dessa saída.

"A bomba B2 é ligada até o nível L1b ser verdadeiro."

Passo 12	Ativar o contator do motor M1[S], ativar o contator da bomba B2.
	Se ("L1b" for verdadeiro) então vá para o Passo 13.

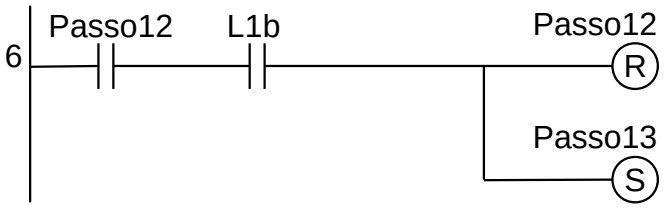

Figura 4.72 – Programa em linguagem Ladder para ao Passo 12.

3) "O motor M1 do agitador é desligado quando o sensor de temperatura T1 for verdadeiro."

Passo 13	Nenhuma ação.
	Se ("T1" for verdadeiro) então vá para o Passo 14.

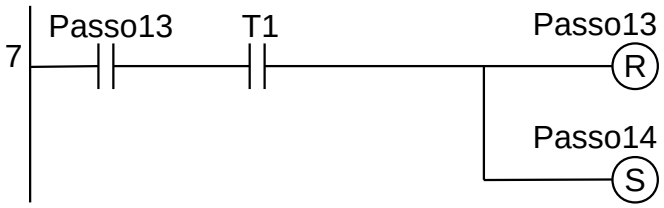

Figura 4.73 – Programa em linguagem Ladder para ao Passo 13.

4) "Aguardar até o sensor de nível L1v ser falso." A memória interna "Tanque1" é feita verdadeira e memorizada para permitir a inicialização do programa do tanque 3.

Passo 14	Desativar o contador do motor M1[R], ativar a memória interna Tanque1[S].
	Se ("L1v" for falso) então vá para o Passo 15.

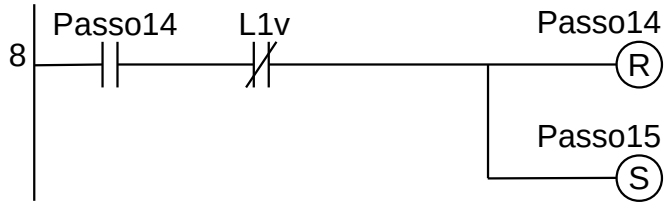

Figura 4.74 – Programa em linguagem Ladder para ao Passo 14.

5) "Reiniciar o processo se a finalização não foi solicitada."

Passo 15	Nenhuma ação.
	Se ("Func" for verdadeiro) então vá para o Passo 11.

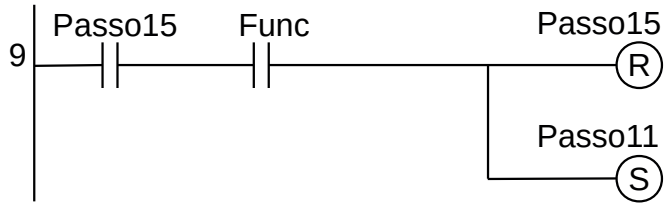

Figura 4.75 – Programa em linguagem Ladder para ao Passo 15.

Automação do tanque 2

1) "A bomba B3 é ligada até o nível L2a ser verdadeiro."

Passo 21	Ativar o contator da bomba B3.
	Se ("L2a" for verdadeiro) então vá para o Passo 22.

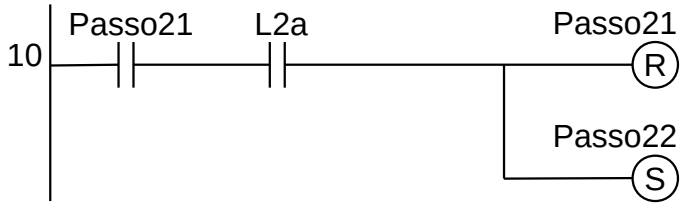

Figura 4.76 – Programa em linguagem Ladder para ao Passo 21.

2) "O motor M2 do agitador é ligado. A resistência de aquecimento Aq é ligado.

O motor M2 do agitador é desligado quando o sensor de temperatura T2 for verdadeiro."

Passo 22	Ativar o contator do motor M2, ativar o contator de aquecimento Aq.
	Se ("T2" for verdadeiro) então vá para o Passo 23

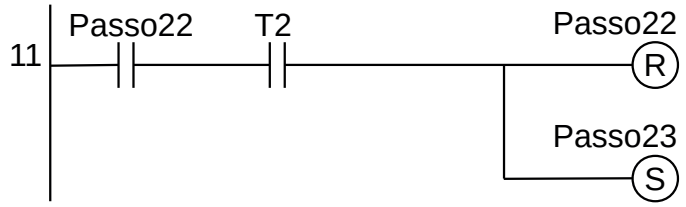

Figura 4.77 – Programa em linguagem Ladder para ao Passo 22.

3) “Aguardar até o sensor de nível L2v ser falso.” A memória interna “Tanque2” é feita verdadeira e memorizada para permitir a inicialização do programa do tanque 3.

Passo 23	Ativar a memória interna Tanque2[S].
	Se (“L2v” for falso) então vá para o Passo 24.

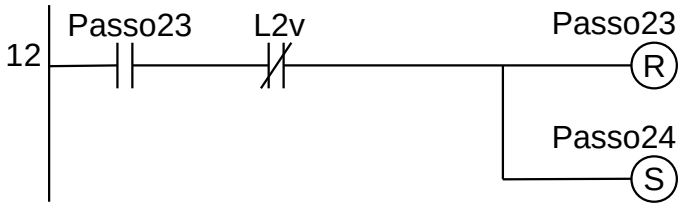

Figura 4.78 – Programa em linguagem Ladder para ao Passo 23.

4) “Reiniciar o processo se a finalização não foi solicitada.”

Passo 24	Nenhuma ação.
	Se (“Func” for verdadeiro) então vá para o Passo 21.

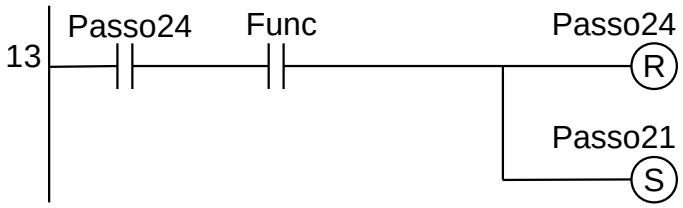

Figura 4.79 – Programa em linguagem Ladder para ao Passo 24.

Automação do tanque 3

1) “Aguardar o término do processo do tanque 1 e do tanque 2. (etapas ‘e’)”

Passo 31	Nenhuma ação.
	Se (“Tanque1” for verdadeiro e “Tanque2” for verdadeiro) então vá para o Passo 32.

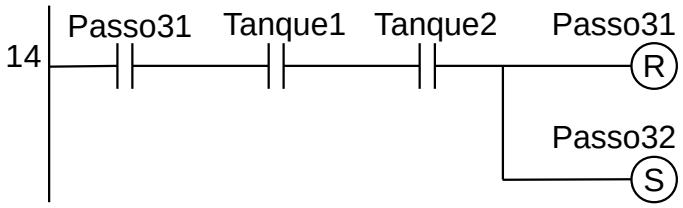

Figura 4.80 – Programa em linguagem Ladder para ao Passo 31.

2) “A bomba B4 é ligada até o nível L1v ser falso.” Desativar as memórias de Tanque1 e Tanque2.

Passo 32	Ativar o contator da bomba B4, desativar as memórias internas Tanque1[R] e Tanque2(R].
	Se (“L1v” for falso) então vá para o Passo 33.

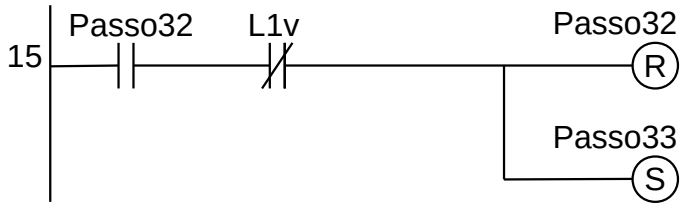

Figura 4.81 – Programa em linguagem Ladder para ao Passo 32.

3) "O motor M3 do agitador é ligado. A bomba B5 é ligada até o nível L2v ser falso."

Passo 33	Ativar o contator do motor M3, ativar o contator da bomba B5. Se ("L2v" for falso) então vá para o Passo 34.

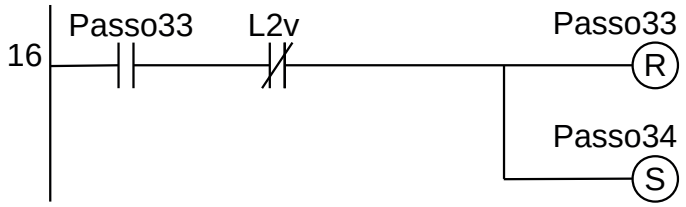

Figura 4.82 – Programa em linguagem Ladder para ao Passo 33.

4) "O motor M3 do agitador é desligado quando o tempo de agitação for de 5 minutos terminar."

Passo 34	Ativar o contator do motor M3, ativar o temporizador Temp1 de 5 minutos. Se ("Temp1" for verdadeiro) então vá para o Passo 35

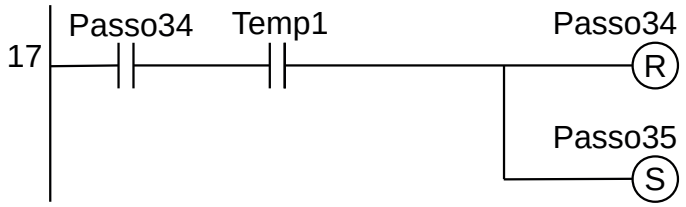

Figura 4.83 – Programa em linguagem Ladder para ao Passo 34.

5) "A bomba B6 é ligada até o nível L3v ser falso."

Passo 35	Ativar o contator da bomba B6. Se ("L3v" for falso) então vá para o Passo 36.

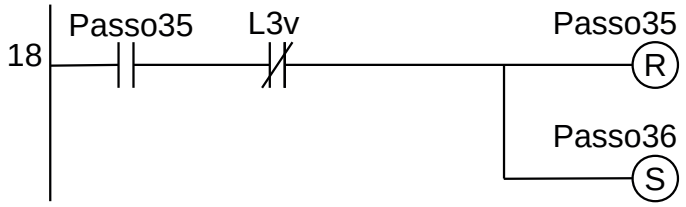

Figura 4.84 – Programa em linguagem Ladder para ao Passo 35.

6) "Reiniciar o processo se a finalização não foi solicitada."

Passo 36	Nenhuma ação. Se ("Func" for verdadeiro) então vá para o Passo 31.

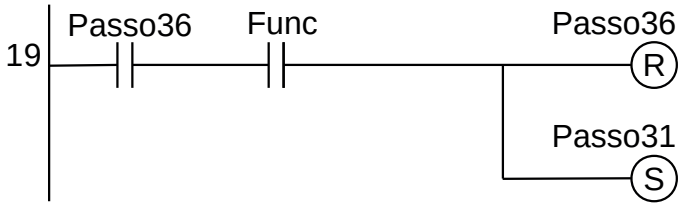

Figura 4.85 – Programa em linguagem Ladder para ao Passo 36.

As saídas são codificadas de acordo com os programas sequenciais.

Ativação dos contatores dos motores de indução das bombas centrífugas.

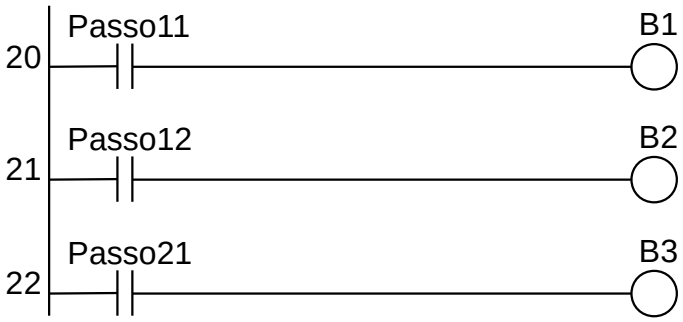

Figura 4.86 – Programa em linguagem Ladder para as saídas B1, B2 e B3.

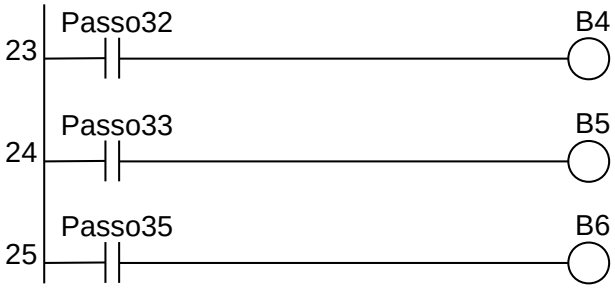

Figura 4.87 – Programa em linguagem Ladder para as saídas B4, B5 e B6.

Ativação dos contatores dos motores de indução dos agitadores.

Figura 4.88 – Programa em linguagem Ladder para as saídas M1, M2 e M3.

Ativação das memórias internas de final de ciclo dos tanques 1 e 2.

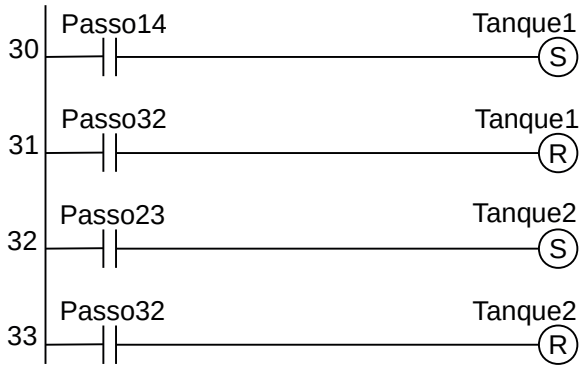

Figura 4.89 – Programa em linguagem Ladder para as variáveis internas Tanque1 Tanque2.

Ativação do contator do aquecedor e do temporizador de 5 minutos.

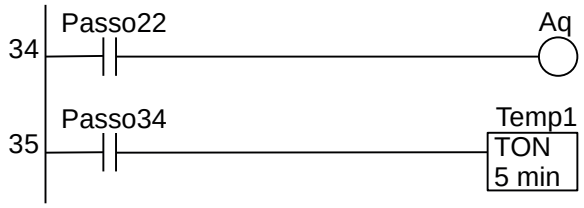

Figura 4.90 – Programa em linguagem Ladder para as saídas Aq e Temp1.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 4.13 e 4.14. A tabela com as variáveis internas pode ser vista na tabela 4.15.

Variável	Etiqueta	Descrição
M01	Passo11	Memória para o passo 11 – tanque 1.
M02	Passo12	Memória para o passo 12 – tanque 1.
M03	Passo13	Memória para o passo 13 – tanque 1.
M04	Passo14	Memória para o passo 14 – tanque 1.
M05	Passo15	Memória para o passo 15 – tanque 1.
M06	Passo21	Memória para o passo 21 – tanque 2.
M07	Passo22	Memória para o passo 22 – tanque 2.
M08	Passo23	Memória para o passo 23 – tanque 2.
M09	Passo24	Memória para o passo 24 – tanque 2.
M10	Passo31	Memória para o passo 31 – tanque 3.
M11	Passo32	Memória para o passo 32 – tanque 3.
M12	Passo33	Memória para o passo 33 – tanque 3.
M13	Passo34	Memória para o passo 34 – tanque 3.
M14	Passo35	Memória para o passo 35 – tanque 3.
M15	Passo36	Memória para o passo 36 – tanque 3.
M16	Trava1	Memória de trava.
M17	Func	Memória para funcionamento.
M18	Tanque1	Memória de fim de ciclo do tanque 1.
M19	Tanque2	Memória de fim de ciclo do tanque 2.
M20	Hab	Memória para habilitação de reinício.
T01	Temp1	Temporizador para ligar (TON) 5 minutos.

Tabela 4.15 – Descrição das variáveis internas.

4.5 – Conclusão

O método de sequência de passos é simples mas deve ser aplicado apenas em sistemas puramente sequenciais sem tomadas de decisão. É possível a aplicação do método em sistema mais complexos, dividindo o programa em vários programas que controlam individualmente partes da máquina ou processo. Outras dificuldades podem ocorrer com sistemas que exigem contagem ou temporização entre eventos. Para estes casos existem outros métodos mais simples.

Uma observação importante é que apenas um único passo de cada programa deve estar ativo. Se a máquina ou processo conter partes que devem atuar simultaneamente, então deve-se dividir o programa em vários pequenos programas que controlam cada parte e desenvolver um programa de gerenciamento desses vários programas.

Capítulo 5
Estrutura de programação por diagrama de tempos

Este método é específico para sistemas sequenciais baseados em intervalos constantes de tempo e sem muita dependência de entradas específicas.

5.1 – Introdução

Alguns processos ou máquinas podem funcionar sem o uso de sensores para informar ao controlador que um determinado evento ocorreu. Esses processos são sequenciais simples e todas as ações são especificadas de acordo com um intervalo de tempo. Uma saída é ligada e desligada em intervalos de tempo regulares e quase sempre repetitivos.

Esse processo de regularidade de tempo pode ser conseguido de modo mecânico com cilindros ou discos que giram em uma velocidade constante. Esses discos ou cilindros apresentam ressaltos, fazendo com que uma parte fique mais baixa que outra parte. Esses dispositivos são chamados de "came mecânico".

Algumas máquinas de costura possuem pontos especiais do tipo bordado linear através de um seletor no painel ou a troca de discos. Quando o disco "came" gira, uma haste passa pelos altos e baixos do disco e movimenta a agulha para o lado, fazendo o desenho do bordado linear. Outra aplicação puramente mecânica é o eixo de comando de válvulas encontrado nos automóveis.

Noutros sistemas o "came" mecânico aciona chaves elétricas do tipo liga e desliga. Um motor síncrono de velocidade constante faz girar um conjunto de discos. Esse tipo pode ser encontrado em algumas máquinas de lavar roupa, máquinas de lavar louça, secadoras de roupas e semáforos. São chamados de temporizadores eletromecânicos por came (em inglês: "Electromechanical Cam Timer").

Em aplicações industriais é utilizado em diversos setores:

- Secadores.
- Fornos profissionais.
- Misturadores.
- Máquinas de lavar roupa e lava-louças industriais.
- Maquinas de calçado.
- Máquinas de confeitaria.
- Máquinas para trabalhar madeira.
- Máquinas de processamento de papel.
- Máquinas de processamento de leite.
- Máquinas de impressão de tela.
- Máquinas de tratamento galvânico.
- Máquinas de embalagem.
- Chaves seccionadoras de média e alta-tensão.
- Ventiladores industriais.

Mais recentemente, os sequenciadores são feitos com placas eletrônicas utilizando circuitos integrados específicos ou microcontroladores que acionam um conjunto de relés eletromecânicos ou

relés de estado sólido. Normalmente essas placas possuem teclas e indicadores visuais para alteração dos parâmetros de temporização.

Alguns fabricantes de controladores lógicos programáveis incorporam um bloco de função que pode realizar a tarefa do sequenciamento de saídas. Mas a maioria dos fabricantes não incorporam essa função e o programador deve desenvolver uma solução com os outros elementos disponíveis. As soluções mais usuais são com o uso de temporizadores para ligar (TON) e manipulação de bits em registradores internos.

5.2 – Histórico

As primeiras referências sobre sequenciadores são as caixinhas de música (em inglês: "music box"). Estas caixas de música são feitas com um cilindro que contém pinos, um conjunto de palhetas metálicas que vibram quando movidas pelos pinos do cilindro, uma mola, engrenagens e um regulador rotativo de velocidade.

No início do século 19, o francês Jacquard desenvolveu um sistema de cartões perfurados e unidos uns aos outros numa sequência para automatizar um tear e assim fazer padrões de desenhos intricados.

No início do século 20 os primeiros semáforos automáticos para controle de tráfego de veículos foram controlados por sistemas de sequenciamento eletromecânicos.

Com o advento de máquinas para uso doméstico nos anos 1950 o sequenciador eletromecânico foi amplamente utilizado.

Os sequenciadores digitais, utilizando circuitos integrados, começam a serem utilizados em meados dos anos 1970.

Os sequenciadores utilizando microprocessador começam a serem utilizados a partir dos anos 1980.

Atualmente utiliza-se circuitos integrados dedicados e microcontroladores.

5.3 – Descrição do método

Para aplicar o método usando temporizadores é necessário:

1. **Entenda o processo.** Verifique os dispositivos de saída podem funcionar sem danos se algum limite de funcionamento for alcançado. Verifique também se o processo ou máquina é totalmente sequencial e sem tomadas de decisão.
2. **Identifique as saídas que são dependentes do tempo.** Faça medições dos tempos necessários para que uma saída fique ativa e a ação seja completada. Verifique os tempos de funcionamento dos dispositivos atuadores sob diversas condições de funcionamento.
3. **Desenhe um diagrama de tempos para as saídas.** O diagrama deve apresentar os tempos em que ocorrem a ativação, ou ligado, e a desativação, ou desligado, de uma determinada saída. Todas as saídas temporizadas são colocadas no mesmo diagrama. Identifique os tempos onde existe transição ligado para desligado e de desligado para ligado. Se o processo é repetitivo, faça o último tempo ser igual ao tempo inicial, também chamado de tempo "zero", que é o momento que a máquina ou processo é iniciado. Um exemplo de diagrama de tempos pode ser visto na figura 5.1, onde "VD" representa a lâmpada verde, "AM" representa a lâmpada amarela e "VM" representa a lâmpada vermelha.

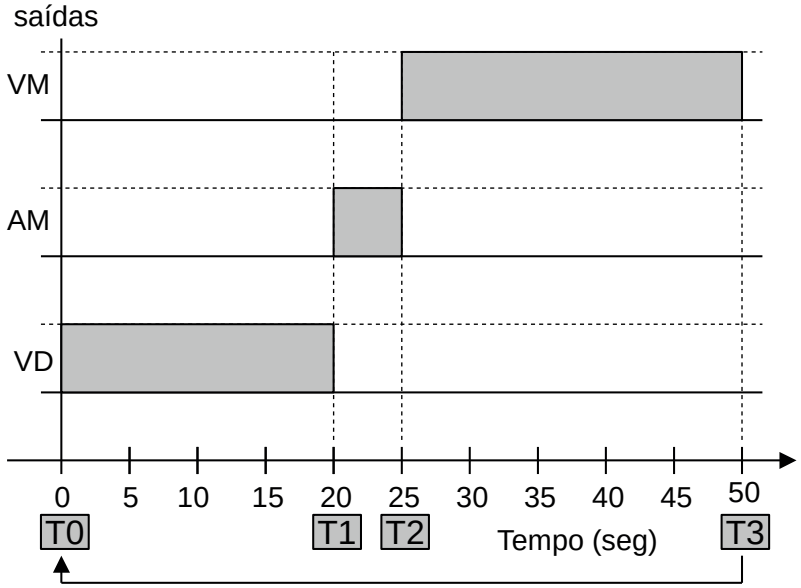

Figura 5.1 – Diagrama de tempos para um semáforo de 3 lâmpadas.

4. **Atribua um temporizador para ligar (TON) para cada tempo quando as saídas estiverem ligadas ou desligadas.** Cada temporizador possui um valor limite atribuído de acordo com os tempos das transições identificadas. Não existe repetição dos valores nos temporizadores. Todos os temporizadores são ativados no tempo "zero" e começam a contar. Cada temporizador vai ter a saída ativa quando atingir o final da contagem pré-programada. Um exemplo de configuração de temporizadores pode ser visto na tabela 5.1.

Etiqueta do temporizador	Valor de tempo (segundos)
T1	20
T2	25
T3	50

Tabela 5.1 – Configuração de temporizadores.

5. **Escreva um programa em linguagem Ladder para ativar todos os valores dos temporizadores.** Se o processo é repetitivo faça o último temporizador desativar todos os temporizadores. Um exemplo de programa em linguagem Ladder para esta estrutura pode ser visto na figura 5.2. Observe que o contato "liga" é utilizado para dar partida ao programa e o contato "T3" pertence ao último temporizador para reinício do processo repetitivo. Se o processo não é repetitivo, esse contato não é utilizado.

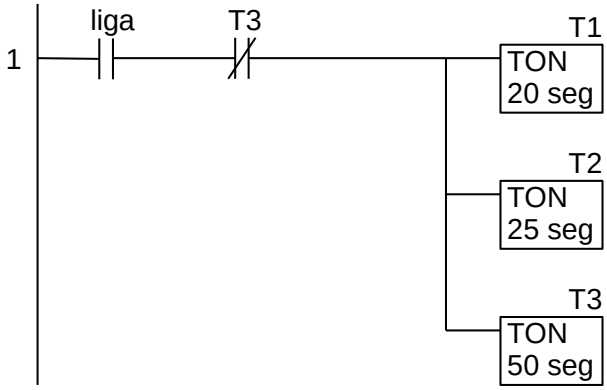

Figura 5.2 – Programa em linguagem Ladder para os temporizadores do semáforo de 3 lâmpadas.

6. **Escreva um programa em linguagem Ladder para ativar e desativar as saídas.** Cada linha de saída do programa em linguagem Ladder possui dois contatos em série: um contato normalmente aberto para "ligar" uma saída e um contato normalmente fechado para "desligar" uma saída. Todos esses contatos estão associados aos temporizadores. Se uma saída for ligada no "tempo zero" (T0) então o contato de "ligar" não é usado ou substituído pelo contato de partida do programa. Um exemplo de programa em linguagem Ladder para esta estrutura pode ser visto na figura 5.3. Observe o contato "liga" na posição de "ligar" para a saída que deve ficar ativada no tempo zero.

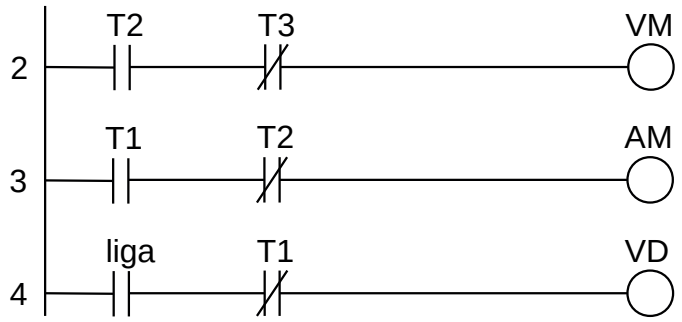

Figura 5.3 – Programa em linguagem Ladder para as saídas do semáforo de 3 lâmpadas.

5.3.1 – Exemplo 1. Semáforo para duas vias

Automatizar um semáforo de trânsito para um cruzamento de uma rua principal com uma rua secundária.

Descrição da máquina ou processo

Um semáforo de trânsito é constituído de 3 lâmpadas: uma lâmpada vermelha, uma lâmpada amarela e uma lâmpada verde. Para um cruzamento de duas ruas, pode-se usar um poste com as três lâmpadas apenas numa rua ou pode-se usar um poste com as três lâmpadas em cada uma das ruas. Nesta instalação não há previsão de semáforo para pedestres ou fase de amarelo piscante em horários de baixo trânsito.

Descrição operacional da máquina ou processo

Considere uma das ruas preferencial, ou seja, o tempo de luz verde para esta rua é maior do que o tempo de luz verde para a rua secundária. Para a rua principal, considere os tempos: verde 32 segundos, amarelo 3 segundos, vermelho 25 segundos. O semáforo segue o padrão americano: verde depois amarelo depois vermelho.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo. Como não existem sensores, somente a tabela de saída é necessária. Para este caso, a tabela das saídas pode ser vista na tabela 5.2.

Parafuso	Etiqueta	Descrição
Q01	PVD	Lâmpada verde da rua principal.
Q02	PAM	Lâmpada amarela da rua principal.
Q03	PVM	Lâmpada vermelha da rua principal.
Q04	SVD	Lâmpada verde da rua secundária.
Q05	SAM	Lâmpada amarela da rua secundária.
Q06	SVM	Lâmpada vermelha da rua secundária.

Tabela 5.2 – Descrição das saídas.

Os tempos de ativação para as saídas são identificados e anotados de acordo as exigências do projeto ou por medições em operação manual da máquina ou processo. Os valores dos tempos para cada saída são mostrados na tabela 5.3.

Etiqueta	Descrição	Tempo (segundos)
PVD	Lâmpada verde da rua principal.	32
PAM	Lâmpada amarela da rua principal.	3
PVM	Lâmpada vermelha da rua principal.	25
SVD	Lâmpada verde da rua secundária.	25
SAM	Lâmpada amarela da rua secundária.	3
SVM	Lâmpada vermelha da rua secundária.	32

Tabela 5.3 – Detalhamento dos tempos para as saídas.

O diagrama de tempos é construído considerando a duração de tempo para cada saída e o sequenciamento entre as saídas. Este diagrama pode ser visto na figura 5.4.

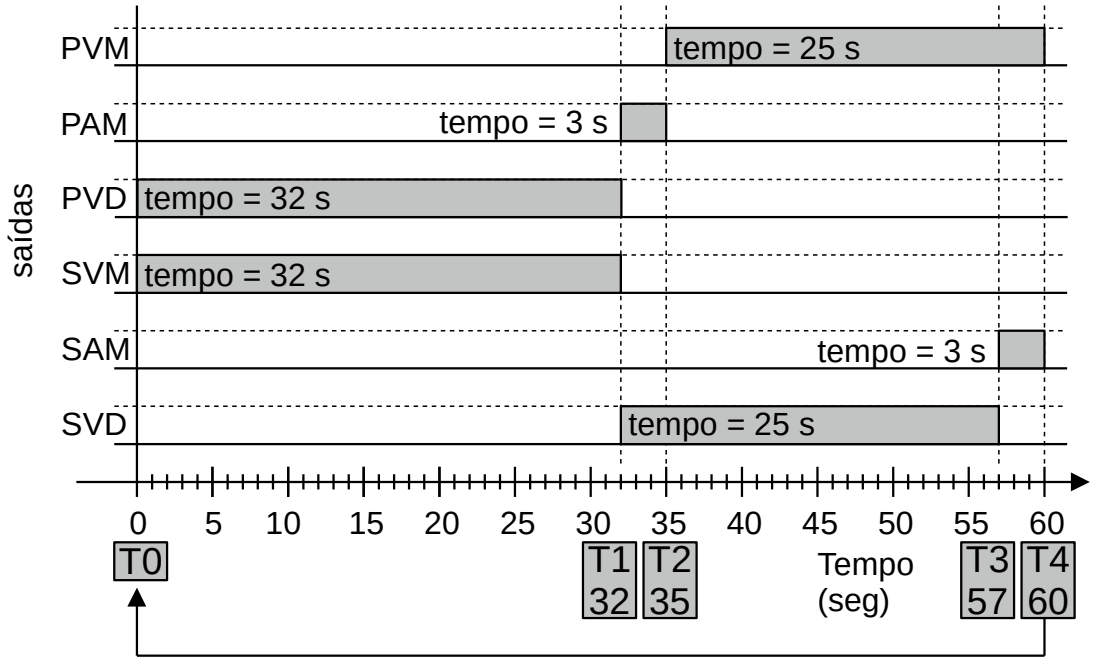

Figura 5.4 – Diagrama de tempos para um semáforo de 6 lâmpadas.

Verifica-se que são necessários 4 temporizadores com tempos de contagem específicos e progressivos. A configuração dos temporizadores pode ser visto na tabela 5.4.

Etiqueta do temporizador	Valor de tempo (segundos)
T1	32
T2	35
T3	57
T4	60

Tabela 5.4 – Configuração de temporizadores.

Para este tipo de programa, onde algumas saídas já iniciam ativadas e é repetitivo, não é necessário um módulo de inicialização. Mas seguindo o método de usar temporizadores, vamos iniciar o programa no "tempo zero" com uma nova modalidade de partida. Na figura 5.5 pode ser visto um temporizador de 1 segundo e sem contato para ativar e desativar. Neste caso, o temporizador é ativado no primeiro ciclo de execução do programa do controlador, mas a saída Tx0 só é ativa após um tempo de 1 segundo. Observe que não existe um contato para ligar e para desligar, ou seja, após o tempo de contagem do temporizador a saída Tx0 continua sempre a ser verdadeira.

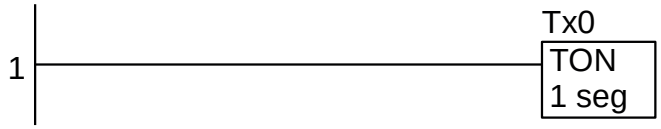

Figura 5.5 – Programa em linguagem Ladder para inicialização.

O programa em linguagem Ladder para ativar todos os 4 temporizadores pode ser visto na figura 5.6. Como o processo é repetitivo, o último temporizador desativa todos os temporizadores. Observe que o tempo de cada temporizador é determinado e invariável.

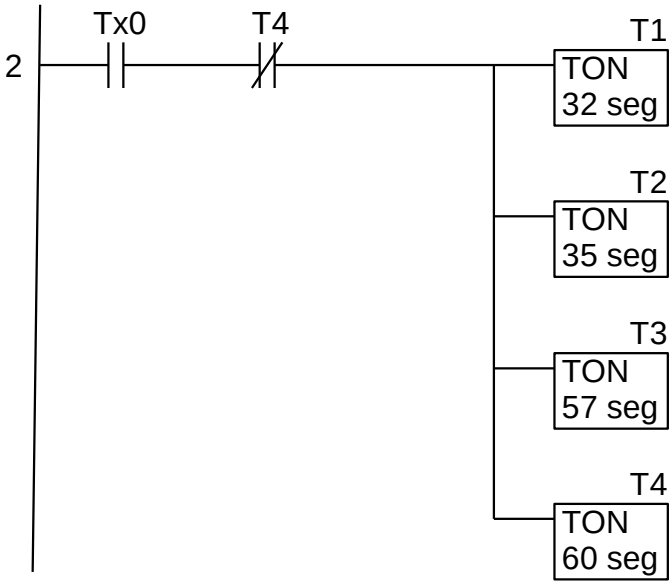

Figura 5.6 – Programa em linguagem Ladder para os temporizadores.

O programa em linguagem Ladder para ativar e desativar as saídas pode ser visto na figura 5.7. Cada linha de saída do programa em linguagem Ladder possui dois contatos em série: um contato

normalmente aberto para ligar uma saída e um contato normalmente fechado para desligar a saída. Todos esses contatos estão associados aos temporizadores. Observe que no “tempo zero” (T0) a lâmpada verde da via principal (PVD) e a lâmpada vermelha da via secundária (SVM) estão ligadas, então usa-se o contato Tx0 para ligar.

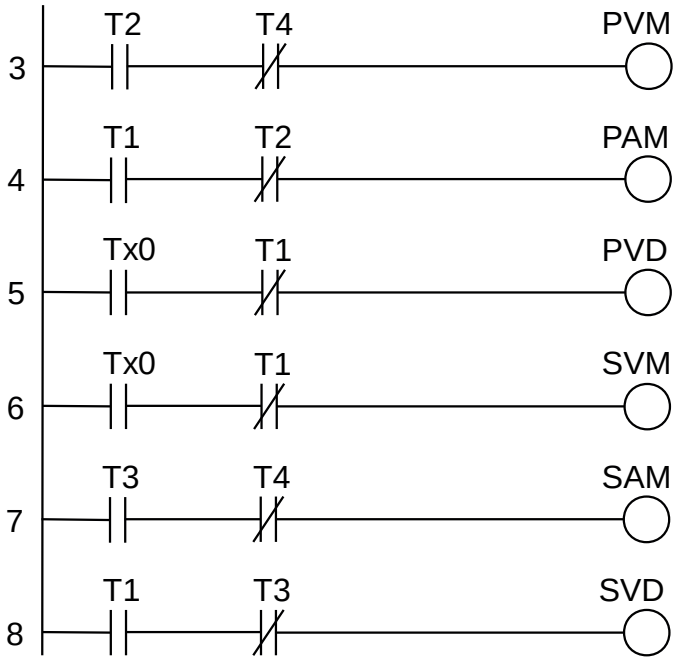

Figura 5.7 – Programa em linguagem Ladder para as saídas.

5.3.2 – Exemplo 2. Sistema “Clean In Place” (CIP)

Uma aplicação típica para um sequenciador é controlar um sistema “Clean In Place” (CIP) para um recipiente de processamento de alimentos, onde um recipiente de processamento deve passar por um ciclo de limpeza para purgá-lo de qualquer matéria biológica entre os ciclos de processamento de alimentos.

Descrição da máquina ou processo

O sistema é composto por tanques que armazenam os produtos de limpeza, bombas centrífugas, válvulas de controle, bicos de dispersão e tubulações.

O CIP em questão possui duas bombas centrífugas:

a) BA – Bomba de abastecimento.

b) BR – Bomba de retorno.

As válvulas de controle são do tipo solenoide e são descritas:

c) VFP – Válvula de fornecimento de pré-enxágue.

d) VFE – Válvula de fornecimento de enxágue.

e) VFK – Válvula de fornecimento de alcalino.

f) VRK – Válvula de retorno de alcalino.

g) VFA – Válvula de fornecimento de ácido.

h) VRA – Válvula de retorno de ácido.

i) VDR – Válvula de drenagem.

Descrição operacional da máquina ou processo

As etapas necessárias para a limpeza do vaso são bem definidas e devem ocorrer sempre na mesma sequência para garantir as condições de higiene. O sistema CIP proposto é executado em 17 etapas:

1. Bombeamento de pré-enxágue (3 min).
2. Pré-enxágue (10 min).
3. Dreno de pré-enxágue (2 min).
4. Bombeamento de alcalino (2 min).
5. Lavagem e circulação alcalino (4 min).
6. Lavar com temperatura (60 min em 75 ºC).
7. Retorno da lavagem ao tanque (2 min).
8. Bombeamento de pré-enxágue (2 min).
9. Pré-enxágue (10 min).
10. Dreno de pré-enxágue (2 min).
11. Bombeamento de ácido (2 min).
12. Lavagem e circulação ácida (4 min).
13. Lavar com temperatura (30 min em 75 ºC).
14. Retorno da lavagem ao tanque (2 min).
15. Bombeamento de enxágue (2 min).
16. Enxaguar (10 min).
17. Dreno de enxágue dreno (3 min).

O sistema é totalmente automático e iniciado a partir de um operador humano pressionar um botão etiquetado como “CIP”. Após o processo de limpeza o sistema se desliga, ou seja, o processo não é repetitivo. Para cada etapa um conjunto de dispositivos de saída são ligados.

- A bomba de abastecimento (BA) deve ficar ligada nas etapas 1, 2, 4, 5, 6, 8, 9, 11, 12, 13, 15 e 16.
- A bomba de retorno (BR) deve ficar ligada em todas as 17 etapas.
- A válvula de fornecimento de pré-enxágue (VFP) deve ficar ligada nas etapas 1, 2, 8 e 9.
- A válvula de fornecimento de enxágue (VFE) deve ficar ligada nas etapas 15 e 16.
- A válvula de fornecimento de alcalino (VFK) deve ficar ligada nas etapas 4, 5 e 6.
- A válvula de retorno de alcalino (VRK) deve ficar ligada nas etapas 5, 6, 7 e 8.
- A válvula de fornecimento de ácido (VFA) deve ficar ligada nas etapas 11, 12 e 13.
- A válvula de retorno de ácido (VRA) deve ficar ligada nas etapas 12, 13, 14 e 15.
- A válvula de drenagem (VDR) deve ficar ligada nas etapas 1, 2, 3, 4, 9, 10, 11, 16 e 17.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entradas é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 5.5.

Parafuso	Etiqueta	Descrição
I01	CIP	Botoeira NA.

Tabela 5.5 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 5.6.

Parafuso	Etiqueta	Descrição
Q01	BA	Contator da bomba de abastecimento.
Q02	BR	Contator da bomba de retorno.
Q03	VFP	Solenoide da válvula de fornecimento de pré-enxágue.
Q04	VFE	Solenoide da válvula de fornecimento de enxágue.
Q05	VFK	Solenoide da válvula de fornecimento de alcalino.
Q06	VRK	Solenoide da válvula de retorno de alcalino.
Q07	VFA	Solenoide da válvula de fornecimento de ácido.
Q08	VRA	Solenoide da válvula de retorno de ácido.
Q09	VDR	Solenoide da válvula de drenagem.

Tabela 5.6 – Descrição das saídas.

O diagrama de tempos é construído considerando a duração de tempo para cada saída e o sequenciamento entre as saídas. Este diagrama pode ser visto na figura 5.8.

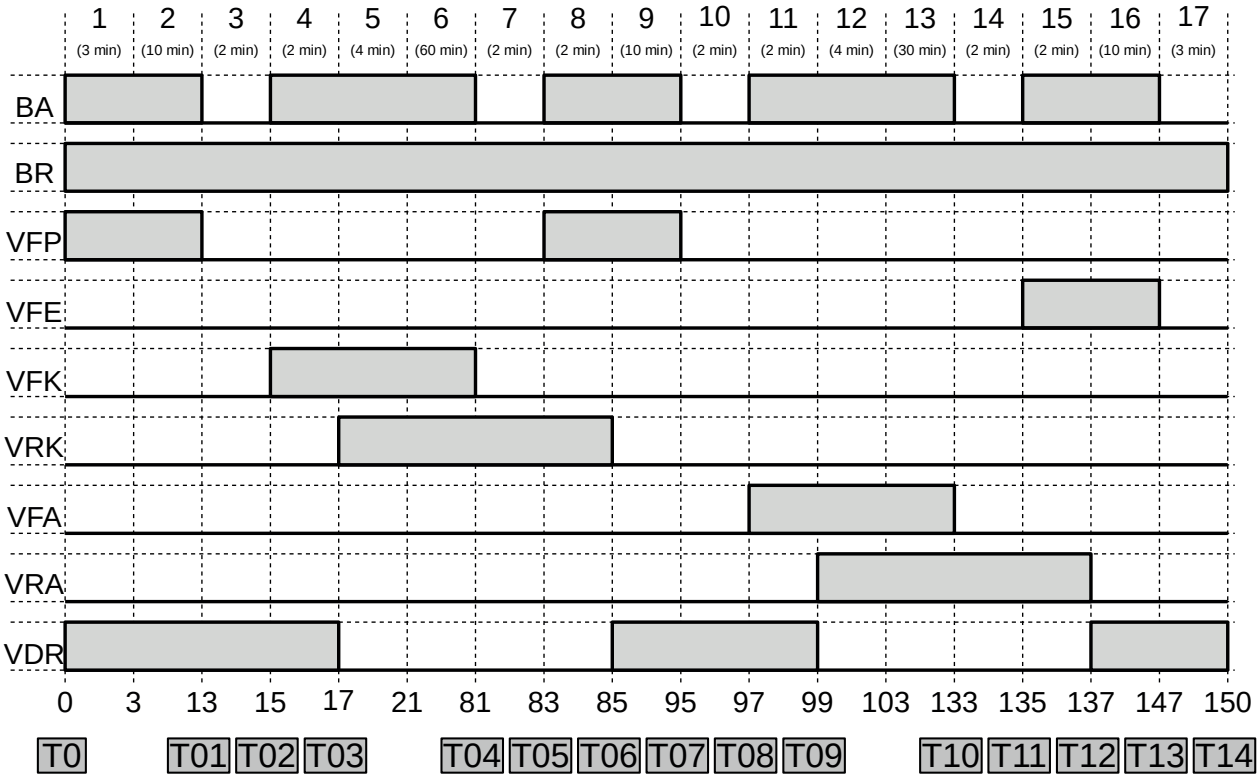

Figura 5.8 – Diagrama de tempos para o processo CIP.

Verifica-se que são necessários 14 temporizadores com tempos de contagem específicos e progressivos. A configuração dos temporizadores pode ser visto na tabela 9.7.

Etiqueta do temporizador	Valor de tempo (minutos)
T01	13
T02	15
T03	17
T04	81
T05	83
T06	85
T07	95
T08	97
T09	99
T10	133
T11	135
T12	137
T13	147
T14	150

Tabela 5.7 – Configuração de temporizadores.

A inicialização do programa se dá quando o operador humano pressionar a tecla “CIP” e deve terminar quando o último contador for verdadeiro. A memória interna etiquetada como “Fun_CIP” é feita verdadeiro para memorizar o evento de funcionamento do processo de CIP. Essa memória interna e feita falsa quando o processo chegar ao final, ou seja, quando o último temporizador terminar a contagem. O programa em linguagem Ladder para esta estrutura pode ser visto na figura 5.9.

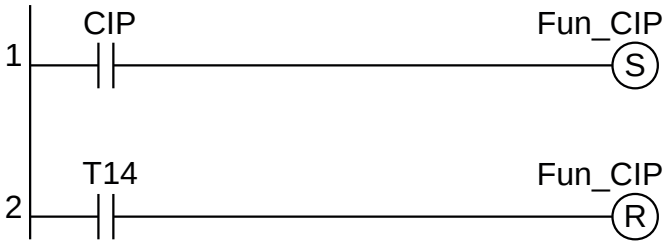

Figura 5.9 – Programa em linguagem Ladder para o processo CIP.

Então, quando a variável “Fun_CIP” for verdadeiro, todos os temporizadores devem iniciar a contagem de acordo com o valor interno de limite de tempo.

Alguns fabricantes de controladores colocam um limite máximo de elementos de saída (bobinas, temporizadores, contadores e outros) que podem ser ativadas simultaneamente. Então uma outra maneira de ativar todos os temporizadores ao mesmo tempo é fazer cada temporizador numa linha de programa e usar o contato de ativação em todos as linhas de programa. O programa em linguagem Ladder para esta estrutura pode ser visto na figura 5.10 e continua na figura 5.11.

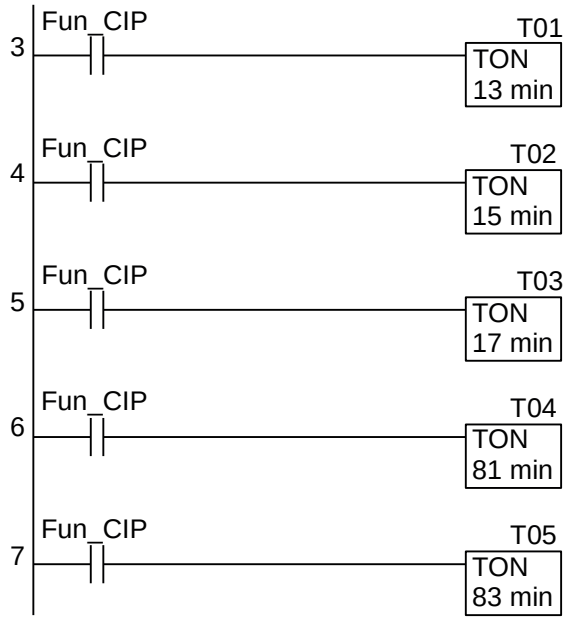

Figura 5.10 – Programa em linguagem Ladder para os contadores (parte 1).

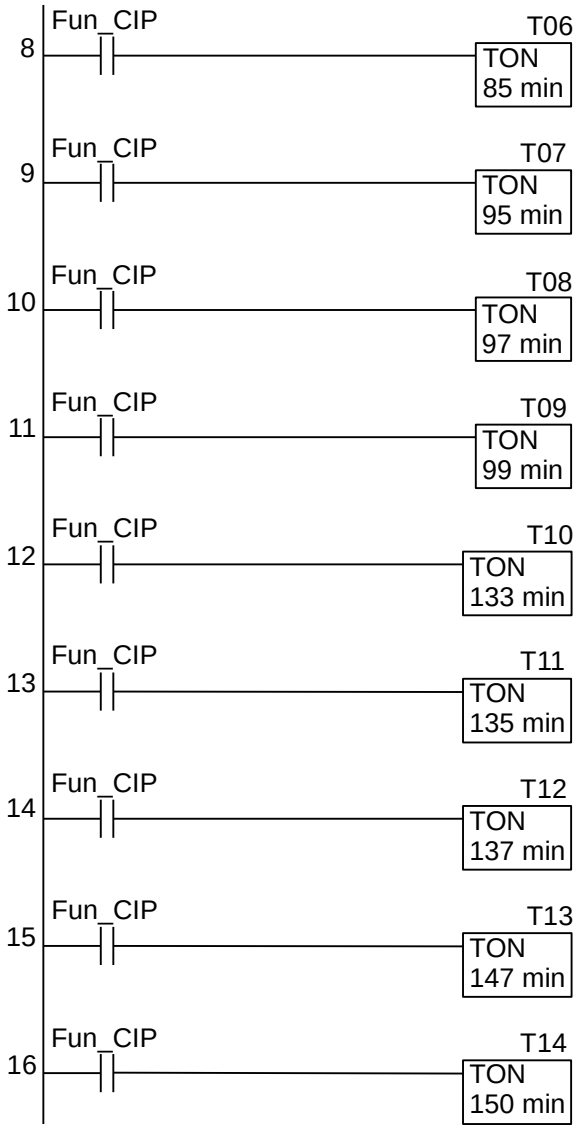

Figura 5.11 – Programa em linguagem Ladder para os contadores (parte 2).

O programa em linguagem Ladder para ativar e desativar as saídas pode ser visto na figura 5.12 e continua na figura 5.13. Cada linha de saída do programa em linguagem Ladder possui dois contatos em série: um contato normalmente aberto para ligar uma saída e um contato normalmente fechado para desligar a saída. Todos esses contatos estão associados aos temporizadores. Observe que no "tempo zero" (T0) as saídas BA, BR, VFP e VDR são ativadas e então usa-se o contato de inicialização "Fun_CIP" no lugar de um contato dos temporizadores. Observe também que as saídas BA, VFP e VDR são ligadas e desligadas mais de uma vez e então usa-se contatos de ligar e desligar em paralelo.

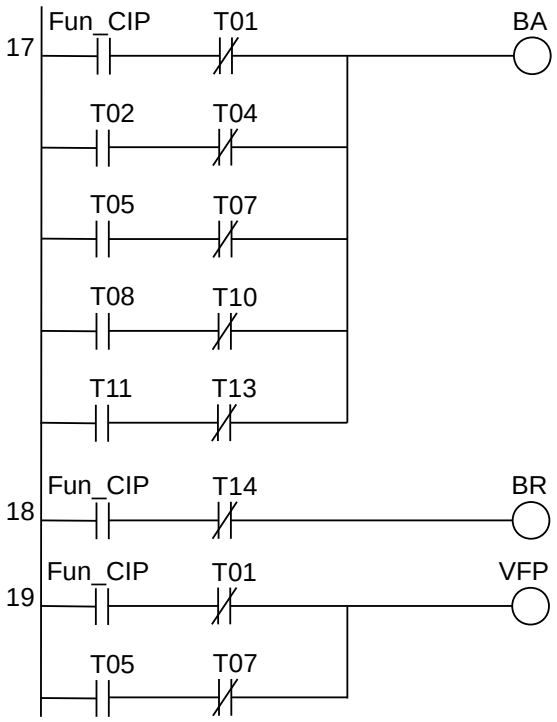

Figura 5.12 – Programa em linguagem Ladder para as saídas (parte 1).

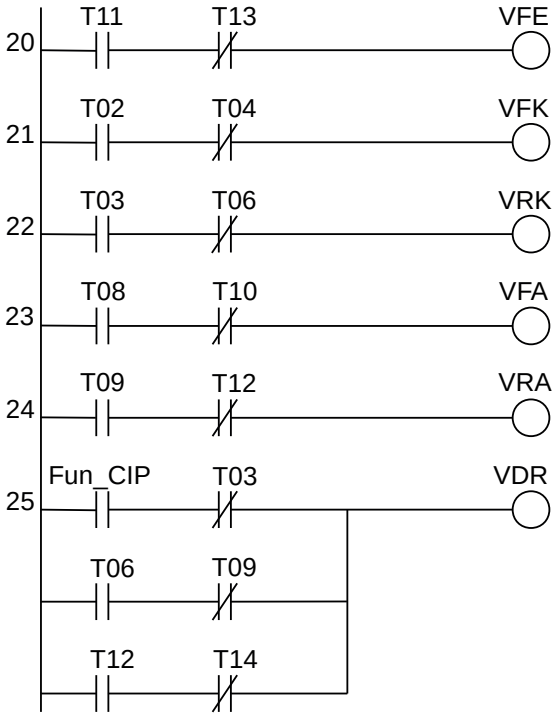

Figura 5.13 – Programa em linguagem Ladder para as saídas (parte 2).

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 5.5 e 5.6. A tabela com as variáveis internas pode ser vista na tabela 5.8.

Variável	Etiqueta	Descrição
T1	T01	Temporizador para ligar (TON).
T2	T02	Temporizador para ligar (TON).
T3	T03	Temporizador para ligar (TON).
T4	T04	Temporizador para ligar (TON).
T5	T05	Temporizador para ligar (TON).
T6	T06	Temporizador para ligar (TON).
T7	T07	Temporizador para ligar (TON).
T8	T08	Temporizador para ligar (TON).
T9	T09	Temporizador para ligar (TON).
T10	T10	Temporizador para ligar (TON).
T11	T11	Temporizador para ligar (TON).
T12	T12	Temporizador para ligar (TON).
T13	T13	Temporizador para ligar (TON).
T14	T14	Temporizador para ligar (TON).
M1	Fun_CIP	Memória interna.

Tabela 5.8– Descrição das variáveis internas.

5.4 – Conclusão

O método por diagramas de tempos é simples e aplicado a sistemas sequenciais simples. Uma das grandes vantagens deste método é a total inexistência de sensores sobre o processo, diminuindo os custos de aquisição de peças físicas como os próprios sensores, cabos, canalização, bornes e pontos de entrada no controlador. Por outro lado a inexistência desses sensores pode comprometer a análise de falhas de atuadores.

No exemplo 1, semáforo, se uma lâmpada apresentar defeito, como queimar, o controlador não tem dispositivos para identificar esse defeito. Os projetistas de semáforos normalmente usam sistema redundante com lâmpadas em paralelo. No exemplo 2, processo de limpeza, a falha em uma eletroválvula pode acarretar em uma limpeza incompleta ou, até mesmo, manter resíduos de produtos de limpeza dentro do tanque de processamento. Em processos de manufatura, com entrada, processamento e saída de produtos, se uma falha de posicionamento ocorrer os produtos podem travar a linha de processamento, impedindo o funcionamento normal e, até mesmo, danificando os dispositivos atuadores.

Variações do método

Variação 1 – Substituir o tempo corrido por intervalos de tempo. Neste caso o término da contagem de tempo do temporizador “n” dispara o início da contagem de tempo do temporizador “n+1”. Na figura 5.14 pode ser visto um exemplo desta técnica. Observe que o temporizador T02 somente vai iniciar a sua contagem após o temporizador T01 terminar a contagem dele. De modo semelhante, o temporizador T03 somente vai iniciar a sua contagem após o temporizador T02 terminar a contagem dele.

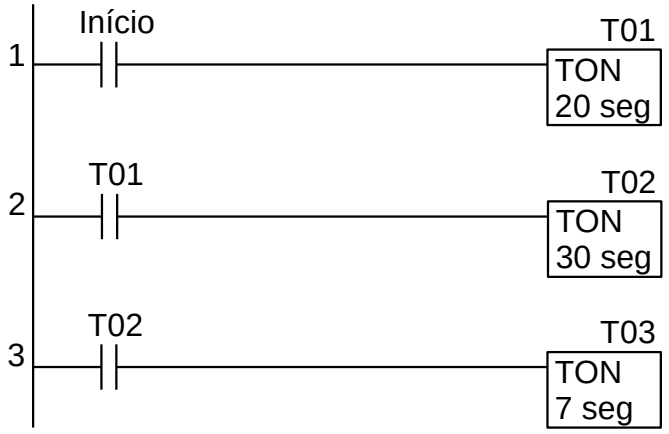

Figura 5.14 – Temporizadores por intervalo de tempo.

Variação 2 – Alteração do tempo dos temporizadores. Em vez de usar valores fixos para contagem de tempo, usa-se valores disponibilizados em registradores onde os tempos são calculados através de fórmulas. Na figura 5.15 pode ser visto um exemplo desta técnica. Observe que o registrador R01 possui o valor numérico inteiro para que o temporizador T01 tenha o limite de contagem estabelecido. Esse registrador pode ter o seu conteúdo alterado por um evento externo, ou uma operação matemática ou movimento de dados entre registradores.

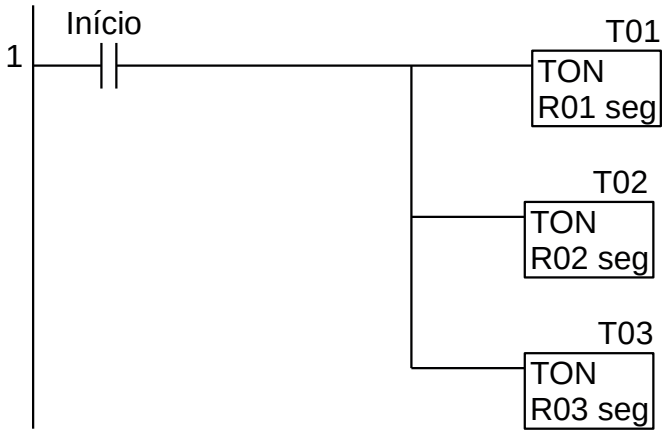

Figura 5.15 – Temporizadores com tempo por registradores.

Variação 3 – Seleção dos temporizadores baseado em uma lógica. É muito útil para sistemas que usam “receitas”, onde para cada lógica de seleção um temporizador fixo é selecionado. Para essa variação funcionar adequadamente, é aconselhável usar em conjunto com a variação 1. Na figura 9.16 pode ser visto um exemplo desta técnica. Observe que os parâmetros A e B são utilizados para selecionar e ativar o primeiro temporizador (T01a de 10 segundos ou T01b de 20 segundos ou T01c de 30 segundos) e o segundo temporizador é ativado para qualquer um dos T01x.

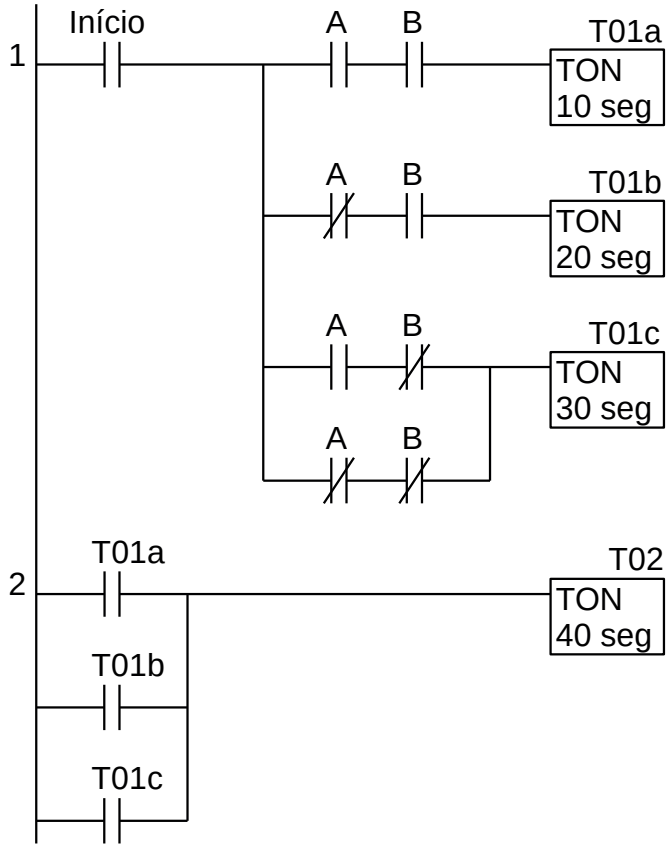

Figura 5.16 – Temporizadores com lógica de seleção.

Capítulo 6
Estrutura de programação por fluxograma

Método amplamente conhecido de programadores e está baseado no fluxo de decisões da combinação de entradas. Não aplicável em sistemas com decisões complexas.

6.1 – Introdução

Um fluxograma é uma representação gráfica de um sistema, processo ou algoritmo de computador, utilizando formas geométricas. Eles são feitos na forma de diagramas simples e fáceis de entender e são usados em vários campos para planejamento, estudo, documentação e programação com objetivo de melhorar e comunicar processos frequentemente complexos.

Fluxogramas, às vezes escritos como "gráfico de fluxo", usam vários elementos gráficos divididos em dois grupos: formas geométricas para eventos de ação e decisão, e linhas com setas para conexão entre as formas geométricas para definir o fluxo e a sequência de ação. As formas geométricas básicas são retângulos, ovais, losangos e círculos. Outras formas geométricas mais específicas são usadas para definir um tipo de etapa especial.

Os fluxogramas podem ser desenhados à mão, como simples gráficos, até sofisticados diagramas desenhados por computador que descrevem várias etapas e caminhos. Eles são um dos diagramas mais comuns e usados por técnicos e não técnicos em vários campos de atividade.

Os fluxogramas podem ser convencionais, usando apenas os símbolos básicos, ou específicos, usando símbolos próprios para cada área. Uma das aplicações mais conhecida, em programação de computadores, usa-se símbolos como fita perfurada, entrada por teclado, disco rígido, e outros elementos da área de computação.

Os fluxogramas mais específicos deram origem a outros diagramas com variações nas formas geométricas e no modo de interligação de uma forma com outra. Às vezes são chamados por nomes mais especializados, como:

- Fluxograma do Processo,
- Mapa do Processo,
- Fluxograma Funcional,
- Mapeamento do Processo de Negócios,
- Modelagem e Notação do Processo de Negócios (BPMN),
- Diagrama do Fluxo do Processo (PFD).

Eles estão relacionados a outros diagramas populares, como:

- Diagramas de Fluxo de Dados (DFDs),
- Diagramas de Atividades de Linguagem de Modelagem Unificada (UML).

6.2 – História

Nas décadas de 1920 e 1930, os fluxogramas começaram a serem usados para documentar processos de negócios.

Em 1921, Frank e Lillian Gilbreth, engenheiros industriais e especialistas em eficiência, apresentaram o "Gráfico de fluxo de processo" à Sociedade Americana de Engenheiros Mecânicos (ASME) numa palestra intitulada "Gráfico de processo: primeiros passos para encontrar a melhor maneira de trabalhar".

No início dos anos 1930, o engenheiro industrial Allan H. Morgensen utilizou as ferramentas de Gilbreth em suas palestras centradas na simplificação do trabalho, ensinando pessoas da área de negócios como deixar seu trabalho mais eficiente em suas Conferências de Simplificação de Trabalho em Nova York.

Na década de 1940, dois alunos de Morgensen, Art Spinanger e Ben S. Graham, difundiram o tema. Spinanger apresentou os métodos de simplificação de trabalho à "Procter and Gamble", onde desenvolveu seu Programa de Mudança de Métodos Deliberados.

Graham, diretor de engenharia na "Standard Register Industrial", adaptou os gráficos de fluxo de processos ao processamento de informações com seu desenvolvimento do fluxograma multifluxo, para apresentar vários documentos e seus relacionamentos.

Em 1947, a ASME adotou um conjunto de símbolos derivado do trabalho original de Gilbreth como o "Padrão ASME: Gráficos de operação e fluxo de processo."

Em 1949, Herman Goldstine e John Von Neumann usaram fluxogramas (originalmente chamado de diagrama) para desenvolver programas de computador. Os fluxogramas de programação originais de Goldstine e von Neumann podem ser encontrados em seu relatório não publicado, "Planejamento e codificação de problemas para um instrumento de computação eletrônico, Parte II, Volume 1" (1947), que é reproduzido nas obras coletadas de von Neumann.

No Japão, Kaoru Ishikawa (1915-1989), definiu os fluxogramas como uma das principais ferramentas de controle de qualidade, junto com ferramentas complementares como o Histograma, Folha de Verificação e Diagrama de Causa e Efeito (também chamado de Diagrama de Ishikawa).

6.3 – Aplicações

Fluxogramas para programação e algoritmos de computador

Como uma representação visual do fluxo de dados, os fluxogramas são úteis para escrever um programa ou algoritmo e explicá-lo a outras pessoas ou colaborar com elas. Você pode usar um fluxograma para explicar a lógica por trás de um programa antes de começar a codificar o processo automatizado. Pode ajudar a organizar o pensamento geral e fornecer um guia para quando chegar a hora de codificar. Mais especificamente, os fluxogramas podem:

- Demonstrar a forma como o código é organizado.
- Visualizar a execução do código dentro de um programa.
- Mostrar a estrutura de um site ou aplicativo.
- Entender como os usuários navegam em um site ou programa.

Frequentemente, os programadores podem escrever pseudocódigo, uma combinação de linguagem natural e linguagem de computador que pode ser lida por pessoas. Isso pode permitir mais detalhes do que o fluxograma e servir como um substituto para o fluxograma ou como uma próxima etapa para o código real.

Os diagramas relacionados usados em software de computador incluem:

- Linguagem de Modelagem Unificada (Unified Modeling Language - UML): Esta é uma linguagem de propósito geral usada em engenharia de software para modelagem.

- Diagramas de Nassi-Shneiderman: usados para programação estruturada de computadores. Nomeado após Isaac Nassi e Ben Shneiderman, que o desenvolveram em 1972 na SUNY-Stony Brook. Também chamado de “Structograms”.
- Gráficos DRAKON: DRAKON é uma linguagem de programação visual algorítmica usada para produzir fluxogramas.

Fluxogramas em outros campos do conhecimento

Além da programação de computadores, os fluxogramas têm muitos usos em muitos campos diversos.

Em qualquer campo:

- Documentar e analisar um processo.
- Padronizar um processo de eficiência e qualidade.
- Comunicar um processo de treinamento ou compreensão por outras partes da organização.
- Identificar gargalos, redundâncias e etapas desnecessárias em um processo e melhorar ele.

Educação:

- Planejar o curso e os requisitos acadêmicos.
- Criar um plano de aula ou apresentação oral.
- Organizar um grupo ou projeto individual.
- Mostrar um processo legal ou civil, como registro de eleitor.
- Planejar e estruturar a escrita criativa, como letras ou poesia.
- Demonstrar o desenvolvimento de personagens para literatura e cinema.
- Representar o fluxo de algoritmos ou quebra-cabeças lógicos.
- Compreender um processo científico, como o ciclo de Krebs.
- Fazer um mapa de um processo anatômico, como a digestão.
- Mapear os sintomas e o tratamento de doenças ou distúrbios.
- Comunicar hipóteses e teorias, como a hierarquia de necessidades de Maslow.

Vendas e Marketing:

- Traçar o fluxo de uma pesquisa.
- Fazer um gráfico de um processo de vendas.
- Planejar estratégias de pesquisa.
- Mostrar fluxos de registro.
- Divulgar políticas de comunicação, como um plano de relações-públicas de emergência.

Negócios:

- Compreender os processos de pedidos e compras.
- Representar as tarefas ou a rotina diária de um funcionário.
- Entender os caminhos que os usuários percorrem em um site ou em uma loja.
- Desenvolver um plano de negócios ou plano de realização do produto.
- Documentar um processo de preparação para uma auditoria, inclusive para conformidade regulatória.
- Documentar um processo de preparação para uma venda ou consolidação.

Fabricação:

- Denotar a composição física ou química de um produto.
- Ilustrar o processo de fabricação do início ao fim.
- Descobrir e resolver ineficiências em um processo de fabricação ou aquisição.

Engenharia:

- Representar fluxos de processo ou fluxos de sistema.
- Projetar e atualizar processos químicos e de plantas.
- Avaliar o ciclo de vida de uma estrutura.
- Fazer um gráfico de um fluxo de engenharia reversa.
- Demonstrar a fase de projeto e protótipo de uma nova estrutura ou produto.

6.4 – Tipos

Diferentes autores descrevem vários tipos de fluxogramas em termos diferentes.

No livro "Critical Incident Management" de 2003, Alan B. Sterneckert listou quatro tipos de fluxogramas populares, estruturados em torno do conceito de controles de fluxo em vez do fluxo em si:

- Fluxogramas de documentos: "têm a finalidade de mostrar os controles existentes sobre o fluxo de documentos através dos componentes de um sistema. … O gráfico é lido da esquerda para a direita e documenta o fluxo de documentos nas várias unidades de negócios."
- Fluxogramas de dados: mostram "os controles que regem os fluxos de dados em um sistema. … Fluxogramas de dados são usados principalmente para mostrar os canais em que os dados são transmitidos através do sistema, em vez de como os controles fluem."
- Fluxogramas do sistema: "mostram o fluxo de dados para e através dos principais componentes de um sistema, como entrada de dados, programas, mídia de armazenamento, processadores e redes de comunicação".
- Fluxogramas do programa: mostram "os controles colocados internamente em um programa dentro de um sistema."

No livro "Microprocessors: Design and Applications" de 1978, Andrew Veronis delineou três tipos de fluxograma com base no escopo e nível de detalhe:

- Fluxograma do sistema: identifica os dispositivos a serem usados.
- Fluxograma geral: Visão geral.
- Fluxograma detalhado: Mais detalhes.

No livro "A Guide for Programmers" de 1978, Marilyn Bohl listou apenas dois:

- Fluxograma do sistema.
- Fluxograma do programa.

No livro "Quality and Process Improvement" de 2001, Mark A. Fryman diferenciou os tipos de várias maneiras, mais de uma perspectiva de negócios do que de uma perspectiva de computador:

- Fluxograma de decisão.
- Fluxograma lógico.
- Fluxograma de sistemas.
- Fluxograma do produto.
- Fluxograma do processo.

Os tipos de fluxograma adicionais definidos por outros incluem:

- Diagrama de raia, também conhecido como Fluxograma de raia: Para delinear quem faz o quê nos processos entre equipes.
- Fluxograma de fluxo de trabalho: para documentar fluxos de trabalho, muitas vezes envolvendo tarefas, documentos e informações em escritórios.
- Fluxograma da Cadeia de Processo Orientada a Eventos (EPC): Para documentar ou planejar um processo de negócios.
- Fluxograma de especificação e linguagem de descrição (SDL): Para fazer um "brainstorm" de algoritmos de computador usando três componentes básicos: definição de sistema, bloco e processo.

Esses diagramas relacionados às vezes também são considerados como tipos de fluxogramas:

- Diagrama de fluxo de dados (DFD): para mapear o fluxo de informações para qualquer sistema ou processo.
- Diagrama de fluxo do processo (PFD), também conhecido como Fluxograma do processo: Para ilustrar as relações entre os principais componentes de uma planta industrial.
- Modelo e notação de processos de negócios (BPMN 2.0): para modelar as etapas de um processo de negócios planejado.

6.5 – Desenhando um fluxograma

Como planejar e desenhar um fluxograma básico.

1. Defina seu propósito e escopo. O que você espera alcançar? Você está estudando as coisas certas com pontos iniciais e finais apropriados para cumprir esse propósito? Seja detalhado o suficiente em sua pesquisa, mas simples em seus gráficos para se comunicar com o público-alvo.
2. Identifique as tarefas em ordem cronológica. Isso pode envolver conversar com os participantes, observar um processo e/ou revisar qualquer documentação existente. Você pode escrever as etapas em forma de nota ou começar um gráfico aproximado.
3. Organize-os por tipo e forma correspondente, como processo, decisão, dados, entradas ou saídas.
4. Desenhe seu gráfico, esboçando à mão ou usando um programa de computador.
5. Confirme seu fluxograma, percorrendo as etapas com as pessoas que participam do processo. Observe o processo para se certificar de que não esqueceu nada importante para o seu propósito.

6.5.1 – Melhores práticas para fazer um fluxograma

Um fluxograma deve ser feito de modo a ser universalmente aceito. Ele deve ser visualmente agradável para outras pessoas.

Se o fluxograma for feito para compartilhar com outra pessoa ou usado em uma apresentação, é aconselhável usar símbolos padrão. Porém, é importante lembrar que a ideia é dar informações de forma fácil de entender. É perfeitamente aceitável usar uma imagem alternativa em vez do símbolo do documento, desde que o público o compreenda.

Algumas coisas que pode ser feito para tornar o fluxograma melhor, incluem usar símbolos do mesmo tamanho, nomear os blocos de decisão, processos e setas, manter uma ordem vertical de disposição dos blocos e usar setas centralizadas nos blocos.

Erros comuns cometidos ao desenhar fluxogramas

Quando se trata de fluxogramas, uma das coisas mais importantes a se considerar é o elemento de clareza e atenção aos detalhes. Os fluxogramas podem variar desde a solução de problemas simples até problemas complexos. No entanto, há uma lista de erros comuns e diretrizes de fluxograma para evitá-los, com os quais deve-se ter cuidado.

O uso de símbolos apropriados

Cada símbolo tem um significado. Embora possa parecer conveniente usar um símbolo de processo para tudo, isso pode confundir o leitor. Símbolos específicos são usados para obter uma melhor compreensão de quais símbolos são relevantes ao ler sobre o que cada objeto trata.

Evitar que a direção do fluxo seja inconsistente

As duas direções de fluxo mais amplamente aceitas são de cima para baixo ou da esquerda para a direita. Esses dois tipos de direções não devem ser misturados no mesmo fluxograma. A consistência realmente importa.

Esquemas de cores excessivos

O fluxograma é projetado para fornecer uma solução para um problema. A última coisa que deve ser feita é perder a mensagem no ruído visual.

Os tamanhos dos símbolos devem ser consistentes

Manter um fluxograma bem proporcionado é vital para evitar uma confusão visual. Como regra geral, certifica-se de que a altura e a largura sejam proporcionais uma à outra e ao restante dos símbolos no fluxograma. Mas isso não é aplicável a objetos intencionalmente pequenos, como conectores.

A necessidade de direcionamento consistente de ramal

Um fluxograma deve ser lógico em todos os aspectos. Uma das áreas às quais não se presta muita atenção é a direção de um ramal. O melhor exemplo para ilustrar esse ponto é com os símbolos de decisão. Idealmente, as condições VERDADEIRAS devem fluir da parte inferior, enquanto as condições FALSAS devem fluir do lado direito.

Opcionalmente a condição FALSA pode fluir para o lado esquerdo, quando usado para repetição simples. Em configurações de múltipla seleção, a condição FALSA flui para baixo e as condições VERDADEIRAS fluem para a direita.

Símbolos e espaçamento do fluxograma

Na maioria das vezes, opta-se por ignorar esse ponto crucial. Para tornar o fluxograma mais profissional, deve-se manter um espaçamento uniforme em torno dos símbolos. A única exceção a essa regra seriam os símbolos de decisão, que requerem espaço extra para acomodar rótulos de ramificação.

Dimensionamento do tamanho do fluxograma

Um dos fatos mais básicos que são esquecidos é o dimensionamento. Muitas vezes, um fluxograma detalhado é redimensionado para caber em apenas uma página. Isso nunca é uma coisa boa. É melhor ter um fluxograma abrangendo várias páginas do que ser espremido em um espaço pequeno,

onde todos os detalhes são ilegíveis. Uma alternativa é criar um fluxograma de alto nível e vários fluxogramas detalhados de cada atividade.

Fluxogramas estendidos

Se o fluxograma estiver conectado a outro fluxograma, em vez de colocá-lo em apenas uma página, é melhor conectá-lo por meio de um nó circular ao fluxograma em uma página diferente.

Definir caminhos alternativos claramente

Em certos fluxogramas, os processos tendem a se bifurcar. Por motivos de clareza, é melhor que se especifique se um ramo precisa ser seguido ou todos eles.

Ter cuidado com os laços

Os processos podem não funcionar para sempre. No entanto, certifique-se de documentar processos que podem ser muito excessivos e que afetam a clareza do fluxograma.

Ser mais descritivo

É sugerido que se use uma nota de rodapé, uma chamada ou até mesmo um documento separado para oferecer mais detalhes para as descrições das etapas do processo que podem precisar de mais detalhes.

Usar uma chave de fluxograma

Uma das melhores práticas de uso de fluxogramas é ter uma chave de fluxograma descrevendo os símbolos que são usados.

Evitar a imprecisão

Ao desenhar fluxogramas, lembre-se de que verificar as etapas do fluxograma é fundamental para evitar quaisquer imprecisões.

Atenha-se a um nível de detalhe

É melhor que se atenha a um certo nível de detalhe, por exemplo, um alto nível, nível médio ou fluxograma detalhado.

Não deixar espaço para qualquer incerteza

Planejar com antecedência significaria evitar erros indesejados. Portanto, certifique-se de fazer perguntas como, “O que acontece a seguir?”, “Há uma decisão tomada agora?” e “A descrição do processo foi concluída?”

6.5.2 – Regras gerais para fluxogramas

1. Todas as formas do fluxograma são conectadas com setas ou “linhas orientadas”.
2. Os símbolos do fluxograma têm um ponto de entrada na parte superior do símbolo, sem outros pontos de entrada. O ponto de saída para todos os símbolos do fluxograma está na parte inferior, exceto para o símbolo de decisão.

3. O símbolo de decisão possui dois pontos de saída; estes podem ser nas laterais ou na parte inferior e um lado.
4. Geralmente, um fluxograma fluirá de cima para baixo. No entanto, um fluxo ascendente pode ser mostrado, desde que não exceda 3 símbolos.
5. Conectores são usados para conectar interrupções no fluxograma. Exemplos são:
   - De uma página para outra página.
   - Do final da página ao topo da mesma página.
   - Um fluxo ascendente de mais de 3 símbolos
6. Sub-rotinas e programas de interrupção têm seus fluxogramas próprios e independentes.
7. Todos os fluxogramas começam com um símbolo Terminal ou Processo Predefinido (para programas de interrupção ou sub-rotinas).
8. Todos os fluxogramas terminam com um terminal ou um laço infinito.

Simbologia

Na tabela 6.1 pode ser visto os símbolos mais comuns usados para construir fluxogramas.

Símbolo do fluxograma	Nome	Descrição
	Símbolo de início / fim	Também conhecido como "Símbolo do Terminador", este símbolo representa os pontos iniciais, finais e resultados potenciais de um caminho. Frequentemente, contém "Início" ou "Fim" na forma.
	Símbolo de processo	Também conhecido como "Símbolo de ação", esta forma representa um processo, ação ou função. É o símbolo mais usado em fluxogramas.
	Símbolo de decisão	Indica uma pergunta a ser respondida - geralmente sim / não ou verdadeiro / falso. O caminho do fluxograma pode então se dividir em diferentes ramos, dependendo da resposta ou das consequências daí em diante.
	Símbolo do conector	Normalmente usado em gráficos mais complexos, este símbolo conecta elementos separados em uma página.
	Símbolo do conector / link fora da página	Frequentemente usado em gráficos complexos, este símbolo conecta elementos separados em várias páginas com o número da página geralmente colocado dentro do símbolo para fácil referência.
	Símbolo de fluxo direcional	A seta é usada para guiar o visualizador ao longo de seu caminho de fluxograma. E embora existam muitos tipos diferentes de pontas de seta para escolher, recomendamos manter uma ou duas para todo o fluxograma. Isso mantém seu diagrama limpo, mas também permite que você enfatize certas etapas em seu processo.

Tabela 6.1 – Símbolos comuns para fluxogramas.

Para a área de computação alguns símbolos específicos foram criados para facilitar o entendimento das diversas tarefas da área. Alguns desses símbolos podem ser vistos na tabela 6.2.

Símbolo do fluxograma	Nome	Descrição
	Símbolo de entrada / saída	Também conhecido como "Símbolo de Dados", esta forma representa os dados que estão disponíveis para entrada ou saída, bem como representa os recursos usados ou gerados. Embora o símbolo da fita de papel também represente entrada / saída, ele está desatualizado e não é mais usado para diagramas de fluxograma.
	Símbolo de documento	Representa a entrada ou saída de um documento, especificamente. Exemplos e entradas são o recebimento de um relatório, e-mail ou pedido. Exemplos de uma saída usando um símbolo de documento incluem a geração de uma apresentação, memorando ou carta.
	Símbolo de sub-rotina	Indica uma sequência de ações que executam uma tarefa específica incorporada em um processo maior. Essa sequência de ações pode ser descrita com mais detalhes em um fluxograma separado.
	Símbolo de dados armazenados	Este é um objeto de armazenamento de dados geral usado no fluxo do processo, em oposição aos dados que também podem ser armazenados em um disco rígido, fita magnética, cartão de memória ou qualquer outro dispositivo de armazenamento.
	Símbolo de armazenamento interno	Esta é uma forma comumente encontrada em fluxogramas de programação para ilustrar as informações armazenadas na memória, em vez de em um arquivo. Essa forma costuma ser chamada de memória de núcleo magnético dos primeiros computadores; ou a memória de acesso aleatório (RAM) como a chamamos hoje.
	Símbolo de entrada manual	Este objeto é representado por um retângulo com o topo inclinado da esquerda para a direita. O objeto de entrada manual significa uma ação em que o usuário é solicitado a fornecer informações que devem ser inseridas manualmente em um sistema.
	Símbolo de preparação	O símbolo de preparação é usado para diferenciar entre as etapas para preparar o trabalho e as etapas que executam uma ação para concluir o trabalho. Também pode ser usado para delinear a configuração de outras etapas que constituem o mesmo processo.

Tabela 6.2 – Símbolos específicos para fluxogramas de computação.

Para a simbologia completa, veja a norma ISO 5807:1985 "Processamento de informações – símbolos de documentação e convenções para dados, fluxogramas de programa e sistema, gráficos de rede de programa e gráficos de recursos de sistema".

6.6 – Estruturas de fluxogramas

Uso dos símbolos

Os elementos do fluxograma são interligadas por linhas orientadas por setas. Existem dois tipos preferenciais de disposição dos elementos do fluxograma: horizontal e vertical. Não há impedimento de se fazer um fluxograma misto, com os blocos e linhas orientadas em qualquer lugar do gráfico, mas isso pode fazer o fluxograma ser visualmente complicado.

Na disposição vertical os elementos são representados de acordo com a figura 6.1, onde o símbolo de início tem a seta conectada na parte inferior, o símbolo de fim tem a seta conectada na parte superior, o símbolo de processo tem a seta de entrada na parte superior e a seta de saída na parte inferior e o símbolo de decisão tem a seta de entrada na parte superior, a seta de saída na parte inferior para a lógica verdadeira (Sim) e a seta de saída na parte direita para a lógica falsa (Não).

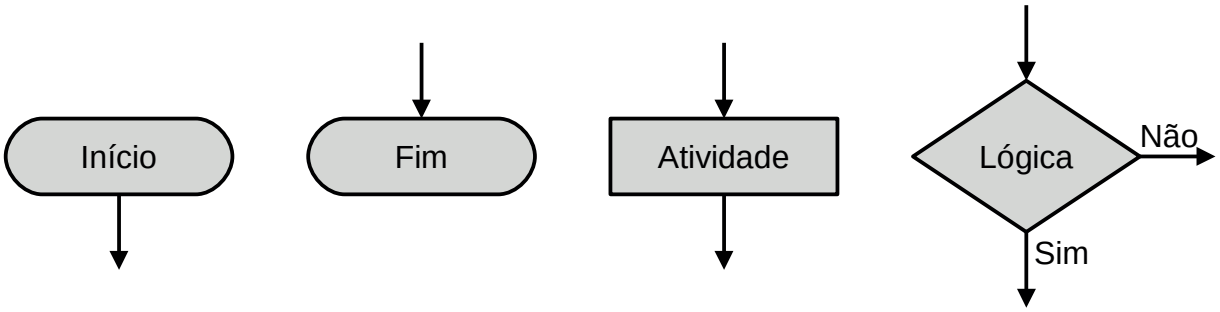

Figura 6.1 – Conexões nos símbolos básicos de fluxograma.

Os símbolos de conectores são usados para tornar o fluxograma mais organizado. Os símbolos de conector interno ( ) são usados para orientar o fluxo da informação dentro da mesma página. Os símbolos de conector externo ( ) são usados para orientar o fluxo da informação de uma página para outra página.

Os símbolos com a linha orientada por seta entrando no símbolo indicam que o fluxo continua no outro símbolo que tem a linha orientada por seta saindo do símbolo. Pode-se utilizar vários símbolos com seta entrando no símbolo mas apenas um símbolo com seta saindo do símbolo. Os conectores internos são identificados com a mesma letra em maiúsculo dentro do símbolo. Os conectores externos são identificados com o número da página dentro do símbolo. Se o símbolo externo ter uma linha orientada por seta saindo, então o número no símbolo indica a numeração da página que o fluxo continua. Se símbolo externo ter uma linha orientada por seta entrando, então o número do símbolo indica a numeração da página que o fluxo teve origem. É aconselhável a combinação de letras e números para diferenciar vários conectores de uma mesma página.

As linhas com setas que entram nos símbolos de conectores internos e externos devem ser preferencialmente pelo lado de cima e as linhas com setas que saem devem ser preferencialmente pelo lado de baixo. Para o conector interno é aceitável a entrada e saída pelos lados da direita e esquerda. Uma disposição de linhas orientadas por seta e símbolos de conectores pode ser visto na figura 6.2.

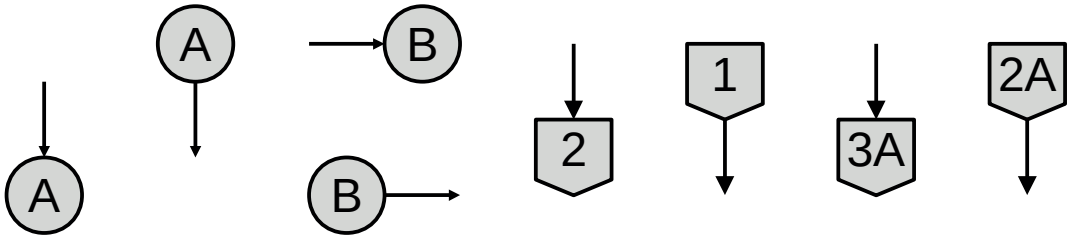

Figura 6.2 – Conexões nos símbolos de conectores de fluxograma.

Uma variação opcional em fluxogramas é o símbolo de conector interno como "convergência em OU". É usado um círculo vazio, sem nenhuma letra ou número interno, e permite uma melhor compreensão da convergência de linhas orientadas por setas. Normalmente utilizado com 2 ou 3 entradas e apenas uma saída. Na figura 6.3 pode ser visto a convergência de duas linhas sem e com o círculo vazio.

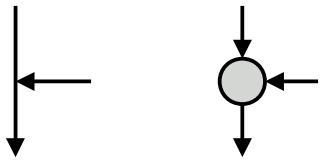

Figura 6.3 – Convergência em OU simples e com símbolo de conector.

A "divergência em OU" é realizada através do símbolo de decisão. A "convergência em E" e a "divergência em E", ou seja, permitindo que mais de um símbolo esteja ativo ao mesmo tempo, é raramente implementada em um mesmo fluxograma. Normalmente usa-se uma barra horizontal composta por 2 linhas sólidas para representar a "convergência em E" e a "divergência em E". Recomenda-se utilizar mais de um fluxograma para representar atividades simultâneas.

Estruturas básicas

Existem três estruturas básicas em fluxogramas:

- Sequência.
- Escolha.
- Repetição.

Uma estrutura é chamada de "**sequência**" quando não existe tomada de decisão e as atividades são executadas em uma sequência única. Um exemplo de fluxograma de sequência pode ser visto na figura 6.4 (a). Neste fluxograma, a "Atividade 1" é executada imediatamente após o início do fluxograma. Depois que a "Atividade 1" terminar, a "Atividade 2" é imediatamente executada. Depois que a "Atividade 2" terminar, a "Atividade 3" é imediatamente executada. E depois o fluxograma termina.

Uma estrutura é chamada de "**escolha**" quando uma tomada de decisão ocorrer devido à análise de uma lógica e uma atividade é executada se a lógica for verdadeira ou a outra atividade é executada se a lógica for falso. Um exemplo de fluxograma de escolha pode ser visto na figura 6.4 (b). Neste fluxograma, a "Atividade 1" é executada imediatamente após o início do fluxograma. Depois que a "Atividade 1" terminar, uma "Lógica" é analisada e se verdadeira (saída Sim) a "Atividade 2" é imediatamente executada. E se a "Lógica" for falsa (saída Não) a "Atividade 3" é imediatamente executada. O fluxograma termina com o término da "Atividade 2" ou o término da "Atividade 3".

Uma estrutura é chamada de "**repetição**" quando uma tomada de decisão ocorrer devido à análise de uma lógica e uma atividade é novamente executada se a lógica for verdadeira ou falsa. O fim da repetição se dá quando a lógica da tomada de decisão for o oposto do elemento de repetição. Um exemplo de fluxograma de repetição pode ser visto na figura 6.4 (c). Neste fluxograma, a "Atividade 1" é executada imediatamente após o início do fluxograma. Depois que a "Atividade 1" terminar, a "Atividade 2" é imediatamente executada. Depois que a "Atividade 2" terminar, uma "Lógica" é analisada e se verdadeira (saída Sim) a "Atividade 3" é imediatamente executada. E se a "Lógica" for falsa (saída Não) a "Atividade 2" é novamente executada. O fluxograma termina com o término da "Atividade 3".

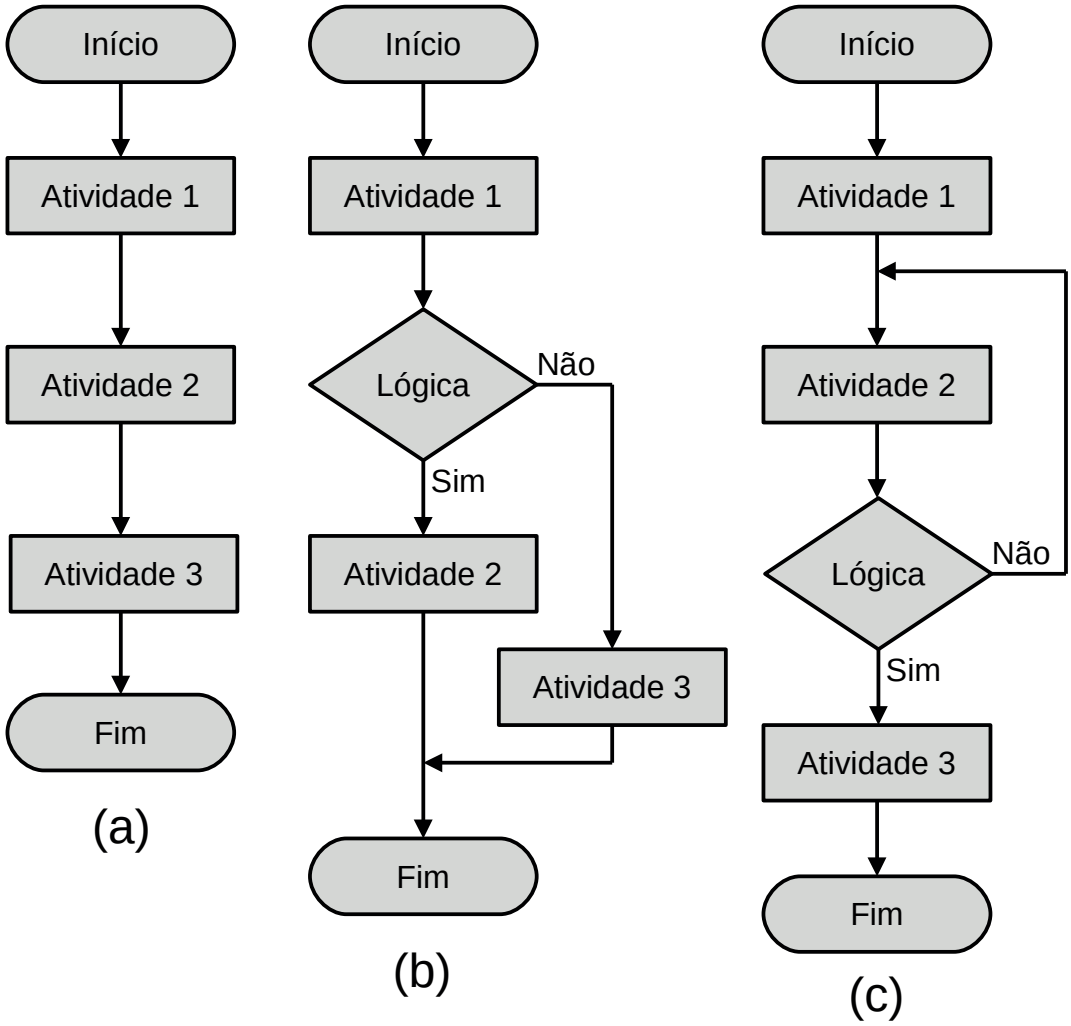

Figura 6.4 – Estruturas básicas de fluxogramas.

6.7 – Aplicação de fluxogramas em controladores industriais

Como os controladores industriais (CLP) fazem as leituras de todas as entradas antes de executar o programa e depois procede à gravação de todas as saídas ao término da execução do programa do usuário, não é necessário o uso dos símbolos gráficos para entrada e saída de variáveis.

Todas as ações referentes a eventos internos e externos são representados dentro dos símbolos de processos (retângulo). Se várias ações são executadas simultaneamente então todas as ações podem ficar dentro de um mesmo símbolo de processo.

Todas as decisões referentes a eventos internos e externos são representados dentro dos símbolos de decisão (losango).

Cada símbolo do fluxograma deve ser etiquetado com um identificador para ser representado na codificação da linguagem do controlador. Uma possibilidade de etiqueta é atribuir um código sequencial para os símbolos. Assim uma sugestão pode ser: B1, B2, B3, … , Bn. Outra possibilidade é diferenciar os blocos retangulares, de processo, como: A1, A2, A3, … , An, e os blocos de decisão como: D1, D2, D3, … , Dn.

6.7.1 – Exemplo 1. O problema do botão e da lâmpada

Ligar e desligar uma lâmpada com um botão tipo "push-button".

O funcionamento é o seguinte:

1) O operador humano aperta o botão e a lâmpada acende imediatamente.
2) O operador humano continua com o dedo no botão e a lâmpada continua acesa.
3) O operador humano retira o dedo do botão e a lâmpada continua acesa.
4) O operador humano aperta, novamente, o botão e a lâmpada apaga imediatamente.
5) O operador humano continua com o dedo no botão e a lâmpada continua apagada.
6) O operador humano retira o dedo do botão e a lâmpada continua apagada.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 6.3.

Parafuso	Etiqueta	Descrição
I01	Botão	Botoeira NA.

Tabela 6.3 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 6.4.

Parafuso	Etiqueta	Descrição
Q01	Lâmpada	Lâmpada comum.

Tabela 6.4 – Descrição das saídas.

Solução por fluxograma

Na figura 6.5 pode ser visto um fluxograma para solução do problema proposto.

Ao iniciar o programa, o bloco B1 é ativado. Considere que a lâmpada esteja apagada.

O bloco B1 analisa a lógica "Botão pressionado". Se for falsa (saída "não") o fluxo permanece no bloco B1. Se for verdadeiro (saída "sim") o fluxo ativa o bloco B2.

O bloco B2 acende a lâmpada. Isso atende à exigência 1.

O bloco B3 analisa a lógica “Botão solto”. Se for falsa (não) o fluxo permanece no bloco B3. Isso atende à exigência 2. Se for verdadeiro (sim) o fluxo ativa o bloco B4.

O bloco B4 analisa a lógica “Botão pressionado”. Se for falsa (não) o fluxo permanece no bloco B4. Isso atende à exigência 3. Se for verdadeiro (sim) o fluxo ativa o bloco B5.

O bloco B5 apaga a lâmpada. Isso atende à exigência 4.

O bloco B6 analisa a lógica “Botão solto”. Se for falsa (não) o fluxo permanece no bloco B3. Isso atende à exigência 5. Se for verdadeiro (sim) o fluxo ativa o bloco B1. Isso atende à exigência 6.

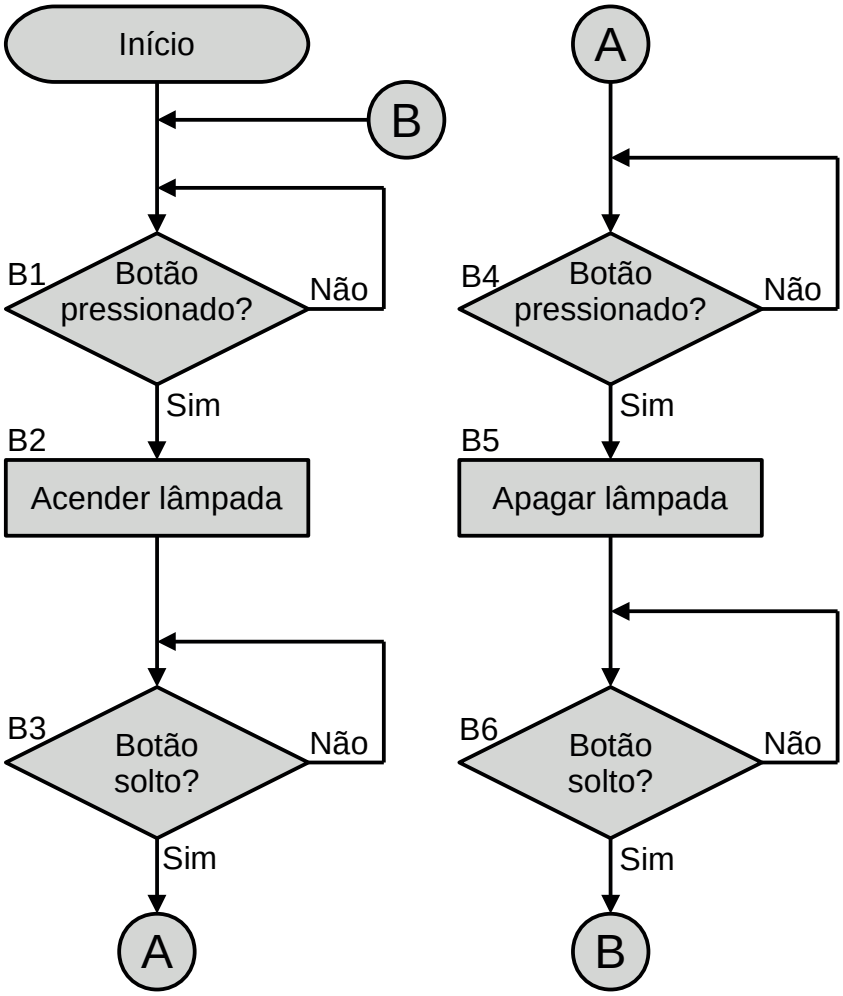

Figura 6.5 – Fluxograma de alto nível para o problema do botão e da lâmpada.

Para fazer a codificação de um fluxograma em um programa para controlador é recomendado que se proceda a uma conversão do fluxograma de alto nível em um fluxograma de baixo nível, referenciando as variáveis internas, variáveis de entrada e variáveis de saída. Na figura 6.6 cada bloco foi identificado com uma etiqueta composta de letras e números de forma sequencial. Essas etiquetas serão usadas para nomear os bits de marcadores auxiliares disponíveis no CLP.

As entradas são identificadas através de etiquetas e serão utilizadas nos blocos de decisão (losangos). Neste exemplo, existe apenas uma entrada e ela pode ser etiquetada como “Botão”. Se o botão não for pressionado, a variável “Botão” tem o valor “falso”. Se o botão for pressionado, a variável “Botão” tem o valor “verdadeiro”.

As saídas são identificadas através de etiquetas e serão utilizadas nos blocos de processo (retângulos). Neste exemplo, existe apenas uma saída e ela pode ser etiquetada como “Lâmpada”. Se a variável “Lâmpada” for “verdadeiro” a lâmpada será ligada. Se a variável “Lâmpada” for “falso” a lâmpada será desligada.

O fluxograma visto na figura 6.6 é uma conversão do fluxograma da figura 6.5 para o modo de baixo nível.

Observe que a análise de “Botão pressionado?” é substituído apenas por “Botão” e a análise de “Botão solto” é substituído por “$\overline{\text{Botão}}$”. A ação “Acender lâmpada” é substituído por “Lâmpada [Set]”, ou seja, é realizada a memorização em “verdadeiro” da saída. A ação “Apagar lâmpada” é substituído por “Lâmpada [Reset]”, ou seja, é realizada a memorização em “falso” da saída.

As saídas de fluxo com etiqueta “Não” retorna no mesmo bloco, e portanto não necessita ser codificada.

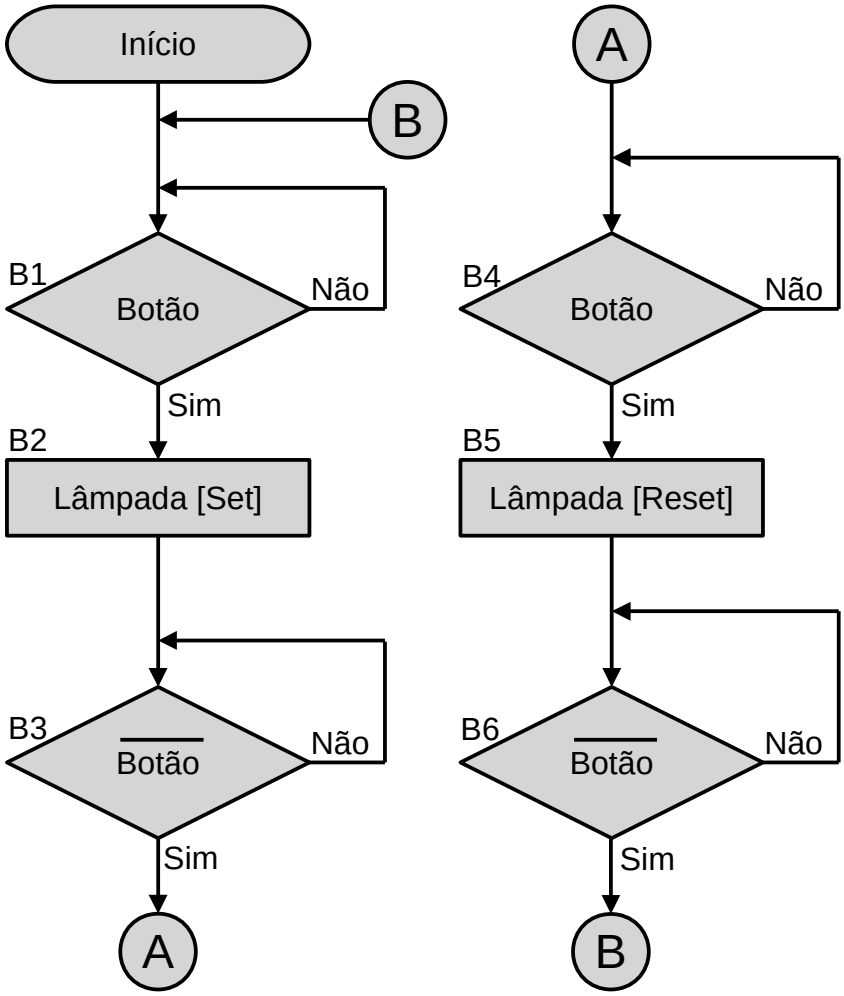

Figura 6.6 – Fluxograma de baixo nível para o problema do botão e da lâmpada.

O programa em linguagem Ladder para este fluxograma pode ser visto na figura 6.7.

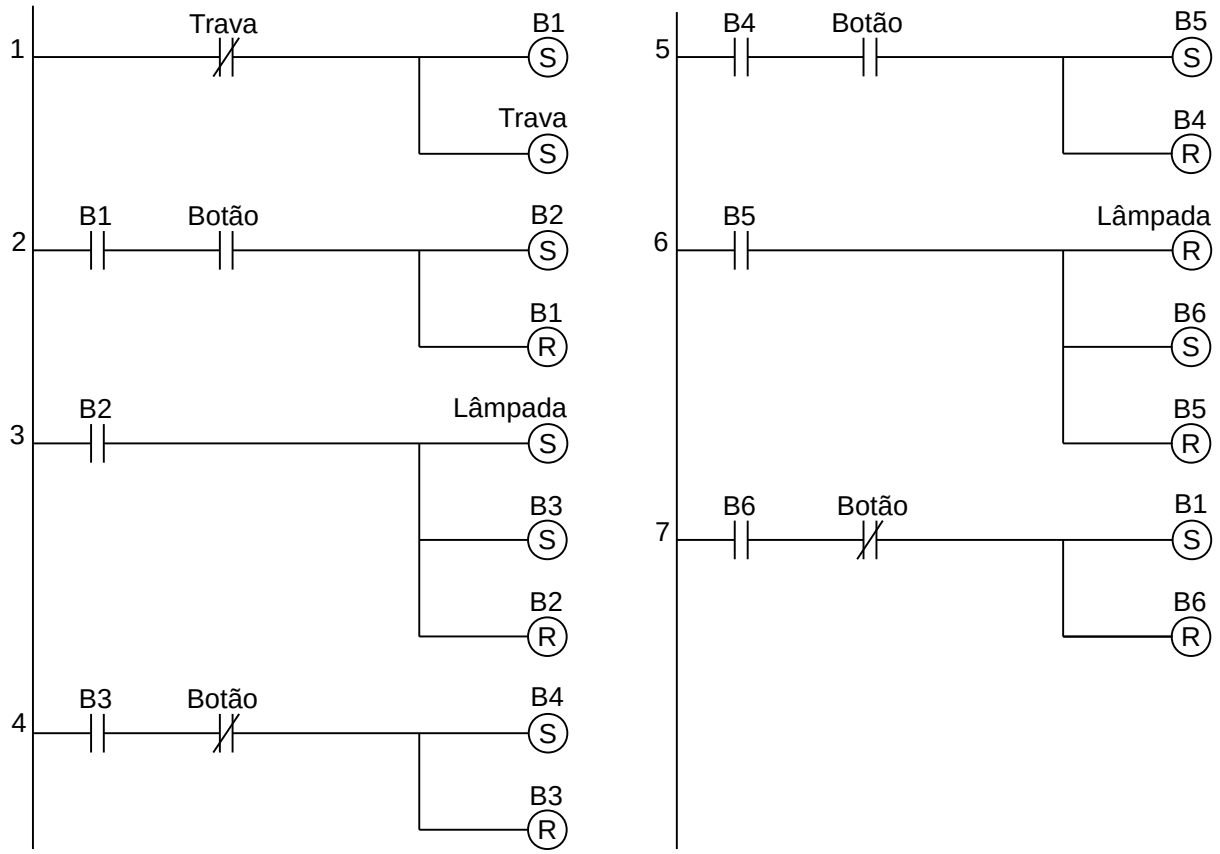

Figura 6.7 – Programa em linguagem Ladder para o fluxograma do exemplo 1.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 6.3 e 6.4. A tabela com as variáveis internas pode ser vista na tabela 6.5.

Variável	Etiqueta	Descrição
M1	B1	Memória para o bloco 1.
M2	B2	Memória para o bloco 2.
M3	B3	Memória para o bloco 3.
M4	B4	Memória para o bloco 4.
M5	B5	Memória para o bloco 5.
M6	B6	Memória para o bloco 6.
M7	Trava	Memória para emulação de "primeiro ciclo".

Tabela 6.5 – Descrição das variáveis internas.

6.7.2 – Exemplo 2. Máquina de etiquetagem por carimbo

Esse exemplo é para mostrar o uso do símbolo de decisão (losango) aplicado a condições de espera, a saída não retorna no próprio bloco, e aplicado a escolha de sequências. A primeira sequência identifica os objetos. A segunda sequência é executada 4 vezes até que um evento externo faça a execução da terceira sequência. Uma máquina de etiquetagem por carimbo é mostrado na figura 6.8.

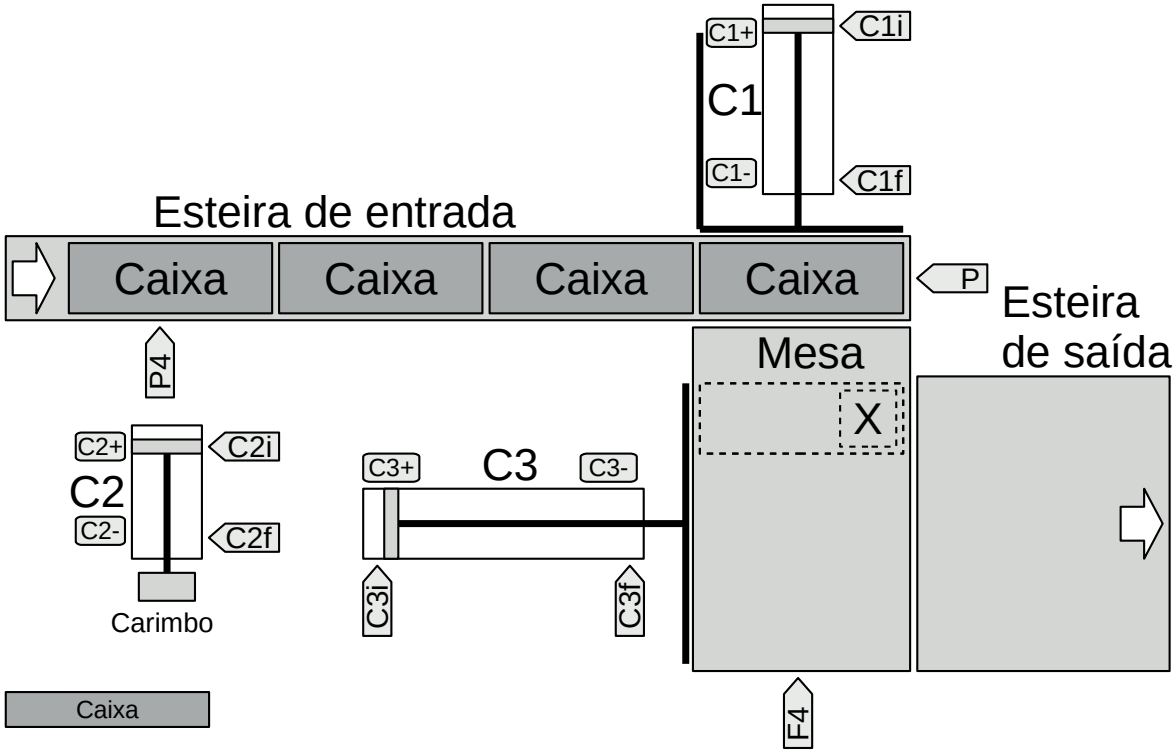

Figura 6.8 – Máquina de etiquetagem por carimbo.

Descrição operacional da máquina ou processo

As caixas a serem etiquetadas chegam através de uma esteira de entrada do tipo acumuladora que possui dois sensores: o sensor “P” identifica que uma caixa está pronta para ser colocada sob o carimbo e o sensor “P4” identifica que existem 4 caixas prontas para serem etiquetadas.

O cilindro pneumático C1 empurra a caixa desde a esteira de entrada até a posição de etiquetagem sob o carimbo, conforme pode ser visto na região tracejada da figura 6.8. O cilindro C1 é acionado pelas eletroválvulas “C1+” para avanço do embolo do cilindro e “C1-” para recuo do embolo do cilindro. O sensor “C1i” identifica o embolo do cilindro C1 totalmente recuado e o sensor “C1f” identifica o embolo do cilindro C1 totalmente avançado.

O cilindro pneumático C2 está montado de modo perpendicular à mesa sobre a região quadrada tracejada indicada com a letra “X”, conforme pode ser visto na figura 6.8. O embolo do cilindro empurra o carimbo em direção à caixa. O cilindro C2 é acionado pelas eletroválvulas “C2+” para avanço do embolo do cilindro e “C2-” para recuo do embolo do cilindro. O sensor “C2i” identifica o embolo do cilindro C2 totalmente recuado e o sensor “C2f” identifica o embolo do cilindro C2 totalmente avançado.

A mesa de etiquetagem é lisa e permite que o cilindro C1 empurre a caixa sob a região de etiquetagem através de outra caixa. Quatro caixas são acumuladas na mesa e são identificadas pelo sensor “F4”.

O cilindro pneumático C3 empurra as 4 caixas desde a mesa de etiquetagem até a esteira de saída. O cilindro C3 é acionado pelas eletroválvulas “C3+” para avanço do embolo do cilindro e “C3-” para recuo do embolo do cilindro. O sensor “C3i” identifica o embolo do cilindro C3 totalmente recuado e o sensor “C3f” identifica o embolo do cilindro C3 totalmente avançado.

Considere que o carimbo deve permanecer em contato com a caixa por 2 segundos, que uma caixa está parada na frente do sensor “P4” se este permanecer ativo por 3 segundos e um alarme.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 6.6.

Parafuso	Etiqueta	Descrição
I01	P4	Sensor óptico de distância.
I02	P	Sensor óptico de distância.
I03	F4	Sensor óptico de distância.
I04	C1i	Sensor magnético de cilindro pneumático.
I05	C1f	Sensor magnético de cilindro pneumático.
I06	C2i	Sensor magnético de cilindro pneumático.
I07	C2f	Sensor magnético de cilindro pneumático.
I08	C3i	Sensor magnético de cilindro pneumático.
I09	C3f	Sensor magnético de cilindro pneumático.

Tabela 6.6 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 6.7.

Parafuso	Etiqueta	Descrição
Q01	C1+	Solenoide de eletroválvula pneumática.
Q02	C1-	Solenoide de eletroválvula pneumática.
Q03	C2+	Solenoide de eletroválvula pneumática.
Q04	C2-	Solenoide de eletroválvula pneumática.
Q05	C3+	Solenoide de eletroválvula pneumática.
Q06	C3-	Solenoide de eletroválvula pneumática.
Q07	Alarme	Lâmpada sinalizadora.

Tabela 6.7 – Descrição das saídas.

Solução por fluxograma

O fluxograma para a máquina de etiquetagem por carimbo pode ser visto na figura 6.9 e 6.10.

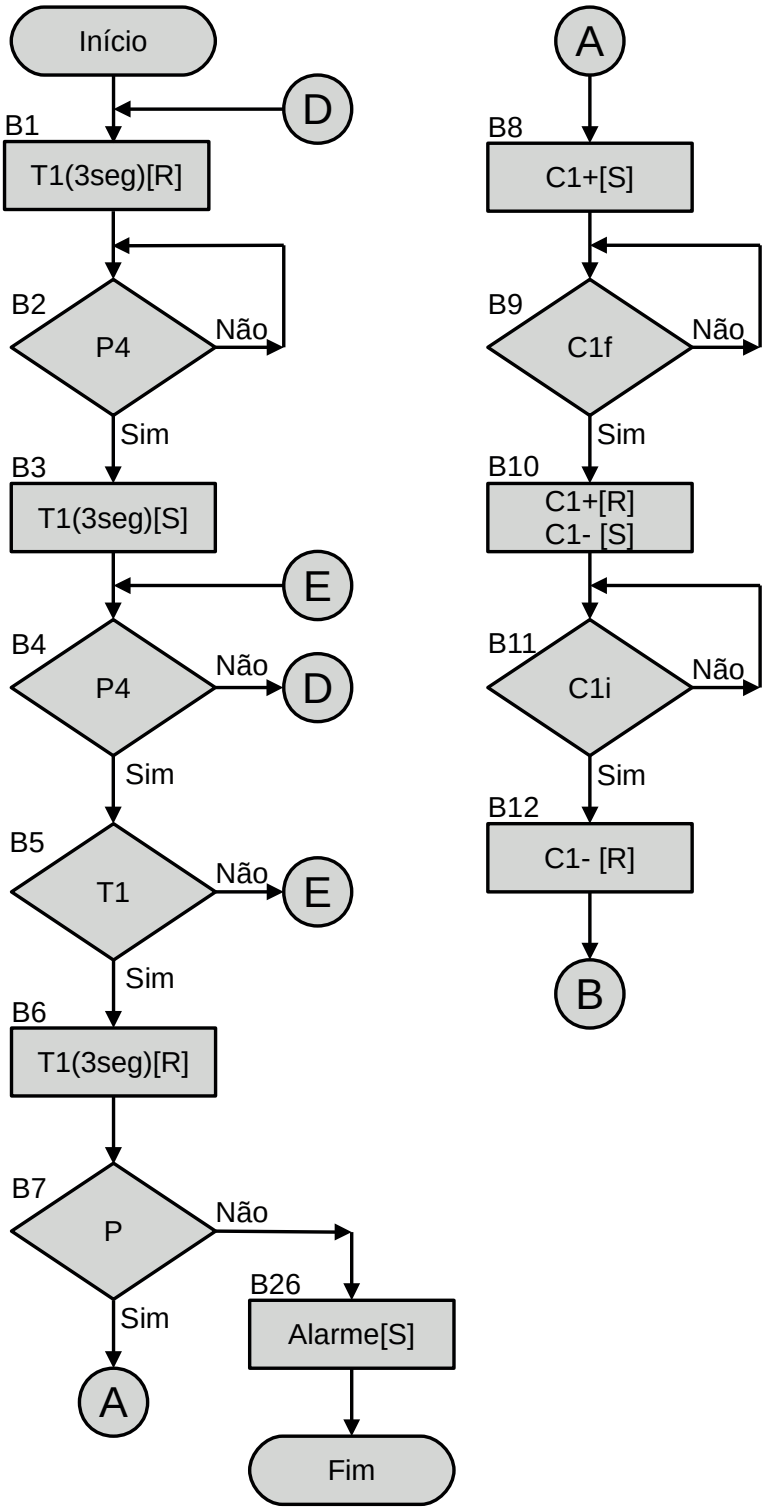

Figura 6.9 – Fluxograma operacional para a máquina de etiquetagem por carimbo (parte 1).

Análise do fluxograma da figura 6.9

A sequência do símbolo de “Início” até o símbolo de conector “A” é para identificar se a quantidade de caixas sobre a esteira acumuladora é de 4 caixas.

Início. Ao dar início ao programa o bloco “B1” é ativado. Este bloco também é ativado através do caminho “D” que pode ter origem no bloco “B4” ou “B25”.

B1. Tipo ação. Desativa a habilitação do temporizador T1 de 3 segundos. E também é realizada a desativação do bloco “B1” e ativação do bloco “B2”. (Isso deve ser feito devido ao retorno do bloco B4)

B2. Tipo decisão. Verifica se uma caixa ativou o sensor “P4”. Se verdadeiro, é feito a desativação do bloco “B2” e a ativação do bloco “B3”, se falso, o bloco “B2” permanece ativo.

B3. Tipo ação. Ativa a habilitação do temporizador T1 de 3 segundos. E também é realizada a desativação do bloco “B3” e ativação do bloco “B4”.

B4. Tipo decisão. Verifica se uma caixa continua ativando o sensor “P4”. Se verdadeiro, é feito a desativação do bloco “B4” e a ativação do bloco “B5”, se falso, é feito a desativação do bloco “B4” e a ativação do bloco “B1”.

B5. Tipo decisão. Verifica se o temporizador T1 terminou a contagem. Se verdadeiro, é feito a desativação do bloco “B5” e a ativação do bloco “B6”, se falso, é feito a desativação do bloco “B5” e a ativação do bloco “B4”.

B6. Tipo ação. Desativa a habilitação do temporizador T1 de 3 segundos. E também é realizada a desativação do bloco “B6” e ativação do bloco “B7”.

B7. Tipo decisão. Verifica se uma caixa ativou o sensor “P”. Se verdadeiro, é feito a desativação do bloco “B7” e a ativação do bloco “B8”, se falso, é feito a desativação do bloco “B7” e a ativação do bloco “B26”.

A sequência do símbolo de conector “A” até o símbolo de conector “B” é para fazer o movimento do cilindro C1 que transfere uma caixa da esteira acumuladora para a mesa de etiquetagem.

B8. Tipo ação. Ativa a saída “C1+” que permite o avanço do embolo do cilindro 1 e transferir a caixa da esteira acumuladora para a mesa de etiquetagem. E também é realizada a desativação do bloco “B8” e ativação do bloco “B9”.

B9. Tipo decisão. Verifica se o sensor “C1f” é verdadeiro, ou seja, se o embolo do cilindro 1 chegou ao final do seu curso, totalmente avançado. Se verdadeiro, é feito a desativação do bloco “B9” e a ativação do bloco “B10”, se falso, o bloco “B9” permanece ativo.

B10. Tipo ação. Desativa a saída “C1+” e ativa a saída “C1-” que permite o recuo do embolo do cilindro 1. E também é realizada a desativação do bloco “B10” e ativação do bloco “B11”.

B11. Tipo decisão. Verifica se o sensor “C1i” é verdadeiro, ou seja, se o embolo do cilindro 1 chegou ao final do seu curso, totalmente recuado. Se verdadeiro, é feito a desativação do bloco “B11” e a ativação do bloco “B12”, se falso, o bloco “B11” permanece ativo.

B12. Tipo ação. Desativa a saída “C1-”. E também é realizada a desativação do bloco “B12” e ativação do bloco “B13”.

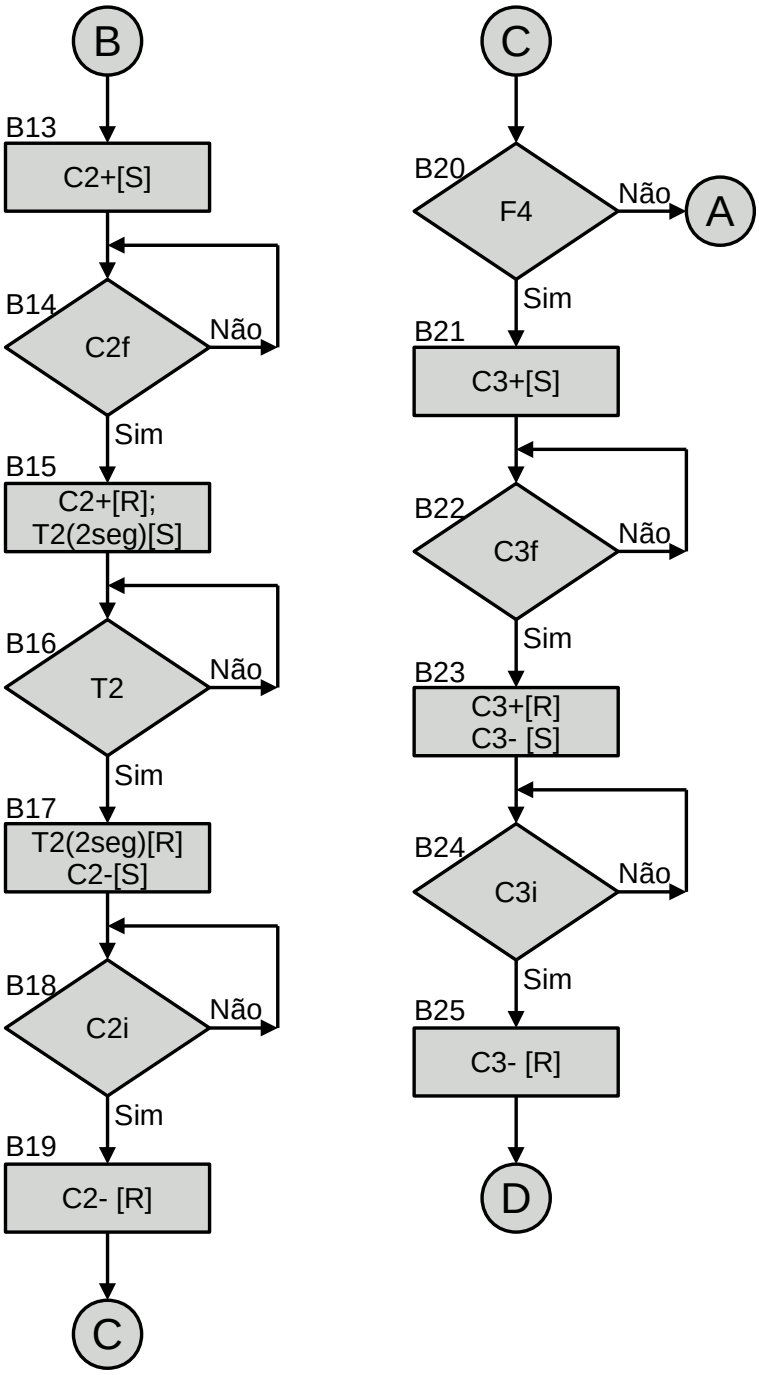

Figura 6.10 – Fluxograma operacional para a máquina de etiquetagem por carimbo (parte 2).

Análise do fluxograma da figura 6.10

A sequência do símbolo de conector “B” até o símbolo de conector “C” é para fazer o movimento do cilindro C2 que realiza a etiquetagem na caixa.

B13. Tipo ação. Ativa a saída “C2+” que permite o avanço do embolo do cilindro 2 e faz a marcação com o carimbo na região de etiquetagem da caixa. E também é realizada a desativação do bloco “B13” e ativação do bloco “B14”.

B14. Tipo decisão. Verifica se o sensor “C2f” é verdadeiro, ou seja, se o embolo do cilindro 2 chegou ao final do seu curso, totalmente avançado. Se verdadeiro, é feito a desativação do bloco “B14” e a ativação do bloco “B15”, se falso, o bloco “B14” permanece ativo.

B15. Tipo ação. Desativa a saída “C2+” e ativa a habilitação do temporizador T2 de 2 segundos. E também é realizada a desativação do bloco “B15” e ativação do bloco “B16”.

B16. Tipo decisão. Verifica se o temporizador T2 terminou a contagem. Se verdadeiro, é feito a desativação do bloco “B16” e a ativação do bloco “B17”, se falso, o bloco “B16” permanece ativo.

B17. Tipo ação. Desativa a habilitação do temporizador T2 de 2 segundos e ativa a saída “C2-” que permite o recuo do embolo do cilindro 2. E também é realizada a desativação do bloco “B17” e ativação do bloco “B18”.

B18. Tipo decisão. Verifica se o sensor “C2i” é verdadeiro, ou seja, se o embolo do cilindro 2 chegou ao final do seu curso, totalmente recuado. Se verdadeiro, é feito a desativação do bloco “B18” e a ativação do bloco “B19”, se falso, o bloco “B18” permanece ativo.

B19. Tipo ação. Desativa a saída “C2-”. E também é realizada a desativação do bloco “B19” e ativação do bloco “B20”.

A sequência do símbolo de conector “C” até o símbolo de conector “D” é para fazer o movimento do cilindro C3 que transfere quatro caixas da mesa de etiquetagem para a esteira de saída.

B20. Tipo decisão. Verifica se uma caixa ativou o sensor “F4”. Se verdadeiro, é feito a desativação do bloco “B20” e a ativação do bloco “B21”, se falso, é feito a desativação do bloco “B20” e a ativação do bloco “B8”.

B21. Tipo ação. Ativa a saída “C3+” que permite o avanço do embolo do cilindro 3 e faz a transferência das 4 caixas da mesa de etiquetagem para a esteira de saída. E também é realizada a desativação do bloco “B21” e ativação do bloco “B22”.

B22. Tipo decisão. Verifica se o sensor “C3f” é verdadeiro, ou seja, se o embolo do cilindro 3 chegou ao final do seu curso, totalmente avançado. Se verdadeiro, é feito a desativação do bloco “B22” e a ativação do bloco “B23”, se falso, o bloco “B22” permanece ativo.

B23. Tipo ação. Desativa a saída “C3+” e ativa a saída “C3-” que permite o recuo do embolo do cilindro 3. E também é realizada a desativação do bloco “B23” e ativação do bloco “B24”.

B24. Tipo decisão. Verifica se o sensor “C3i” é verdadeiro, ou seja, se o embolo do cilindro 3 chegou ao final do seu curso, totalmente recuado. Se verdadeiro, é feito a desativação do bloco “B24” e a ativação do bloco “B25”, se falso, o bloco “B24” permanece ativo.

B25. Tipo ação. Desativa a saída “C3-”. E também é realizada a desativação do bloco “B25” e ativação do bloco “B1”.

B26. Tipo ação. Ativa a saída de “Alarme”. E também é realizada a desativação do bloco “B26”.

Fim. O programa termina e fica inoperante até o operador da máquina reiniciar o controlador.

Codificação do fluxograma na linguagem Ladder

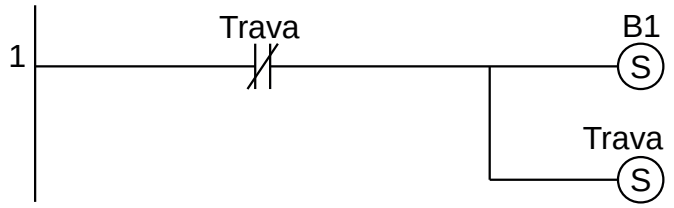

Figura 6.11 – Programa em linguagem Ladder para o bloco de “Início”.

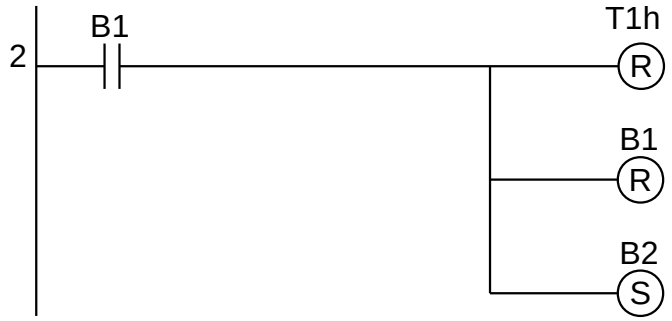

Figura 6.12 – Programa em linguagem Ladder para o bloco B1.

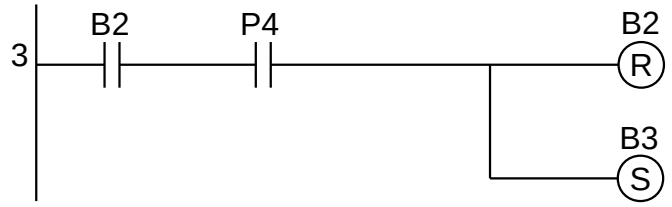

Figura 6.13 – Programa em linguagem Ladder para o bloco B2.

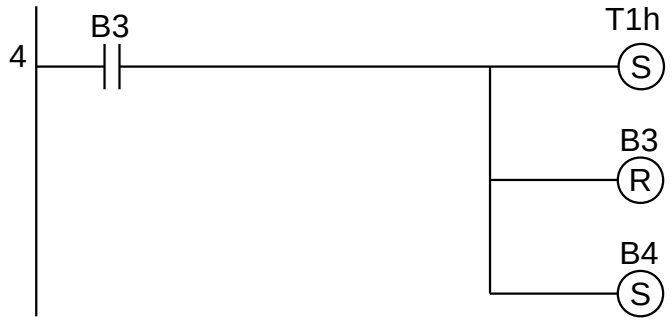

Figura 6.14 – Programa em linguagem Ladder para o bloco B3.

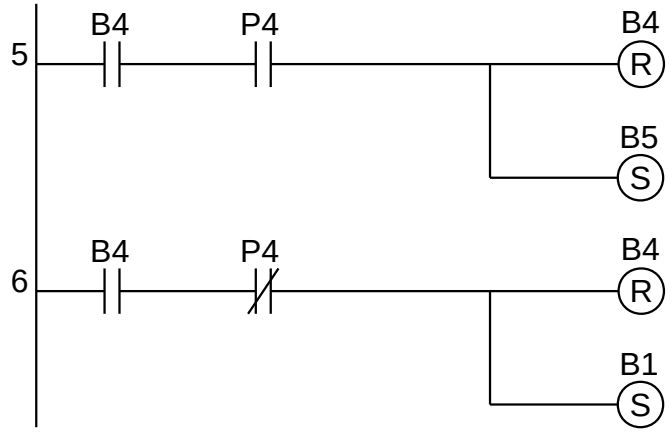

Figura 6.15 – Programa em linguagem Ladder para o bloco B4.

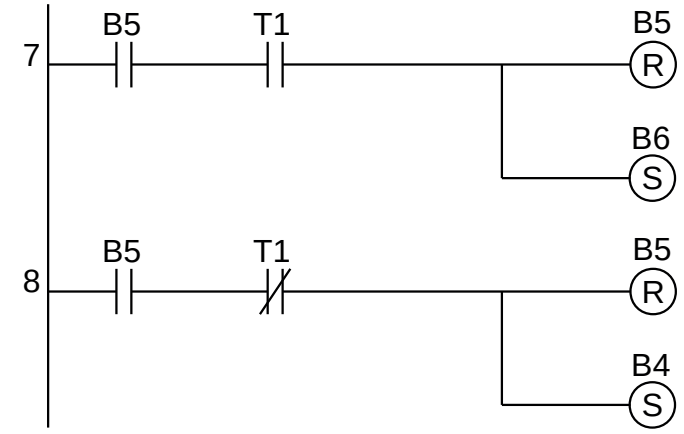

Figura 6.16 – Programa em linguagem Ladder para o bloco B5.

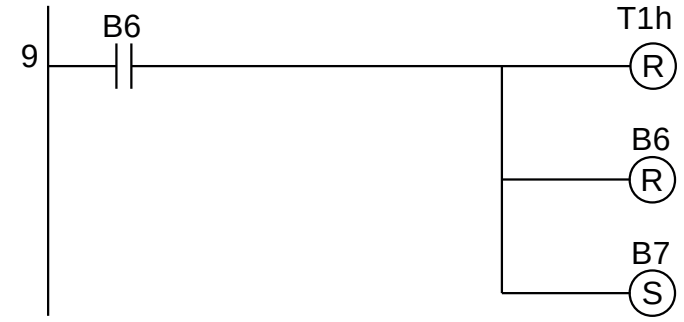

Figura 6.17 – Programa em linguagem Ladder para o bloco B6.

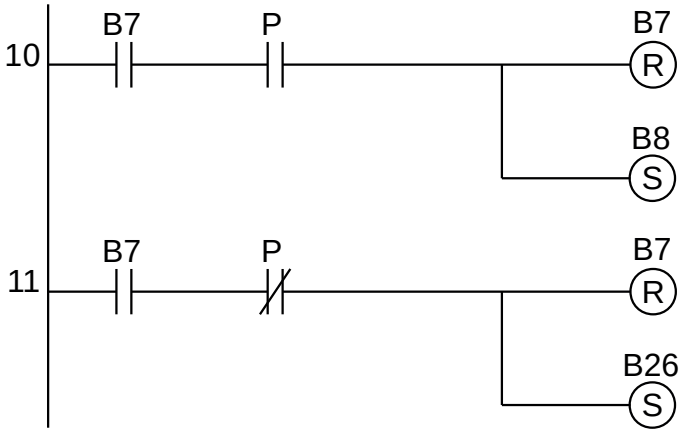

Figura 6.18 – Programa em linguagem Ladder para o bloco B7.

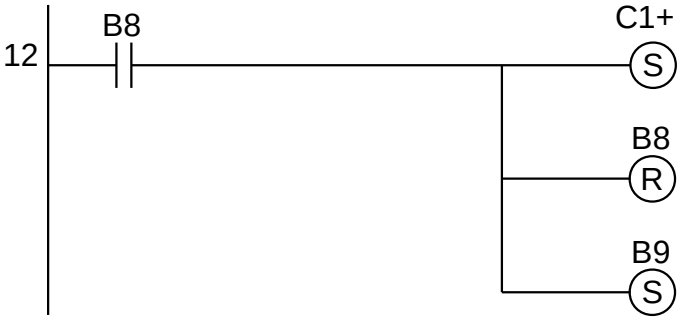

Figura 6.19 – Programa em linguagem Ladder para o bloco B8.

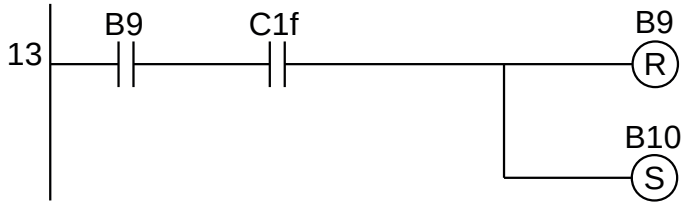

Figura 6.20 – Programa em linguagem Ladder para o bloco B9.

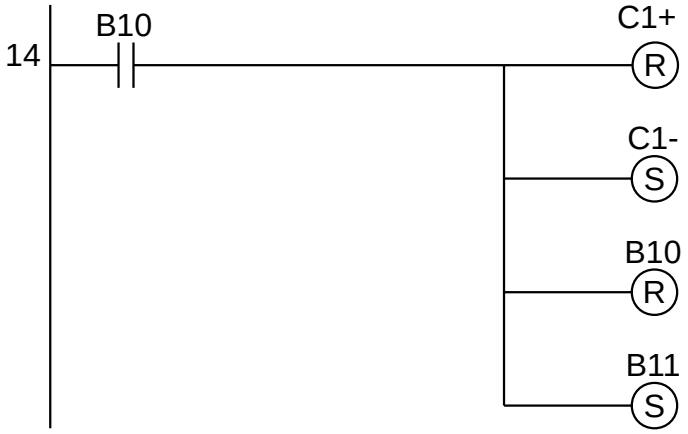

Figura 6.21 – Programa em linguagem Ladder para o bloco B10.

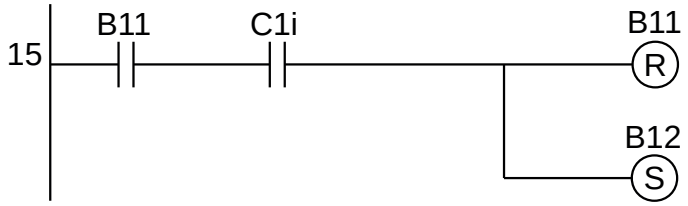

Figura 6.22 – Programa em linguagem Ladder para o bloco B11.

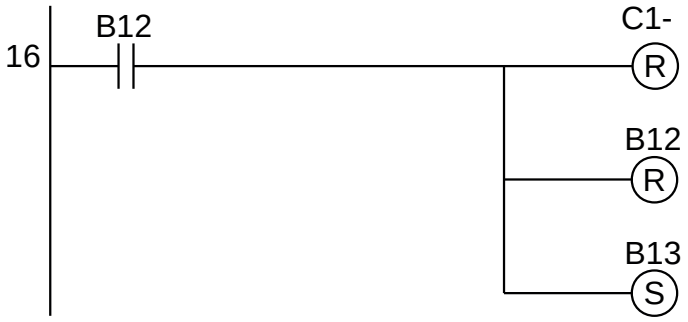

Figura 6.23 – Programa em linguagem Ladder para o bloco B12.

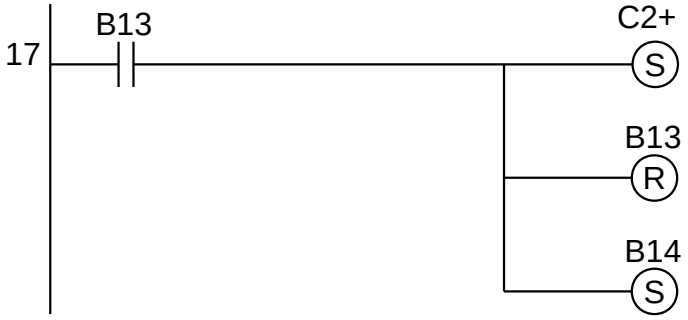

Figura 6.24 – Programa em linguagem Ladder para o bloco B13.

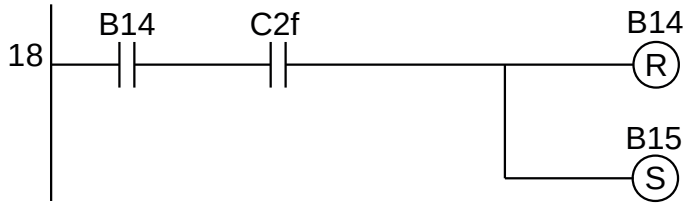

Figura 6.25 – Programa em linguagem Ladder para o bloco B14.

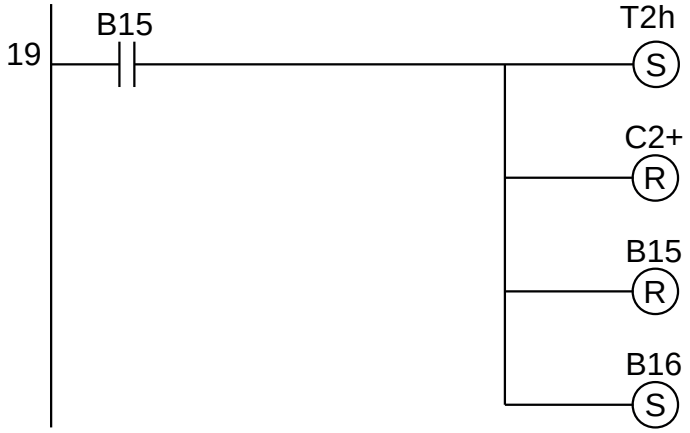

Figura 6.26 – Programa em linguagem Ladder para o bloco B15.

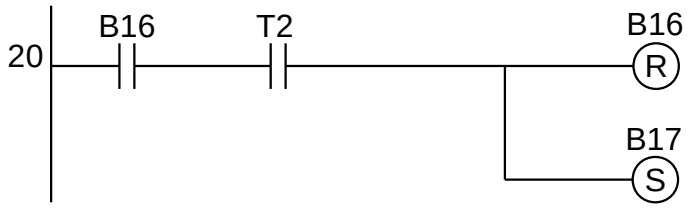

Figura 6.27 – Programa em linguagem Ladder para o bloco B16.

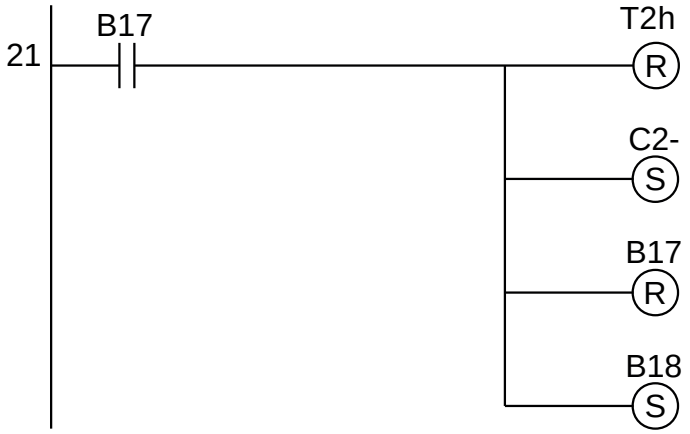

Figura 6.28 – Programa em linguagem Ladder para o bloco B17.

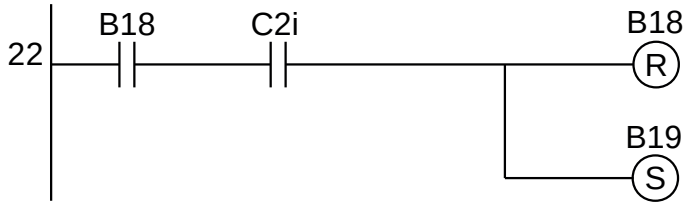

Figura 6.29 – Programa em linguagem Ladder para o bloco B18.

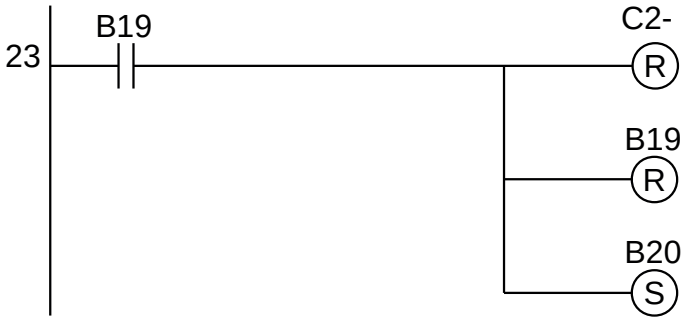

Figura 6.30 – Programa em linguagem Ladder para o bloco B19.

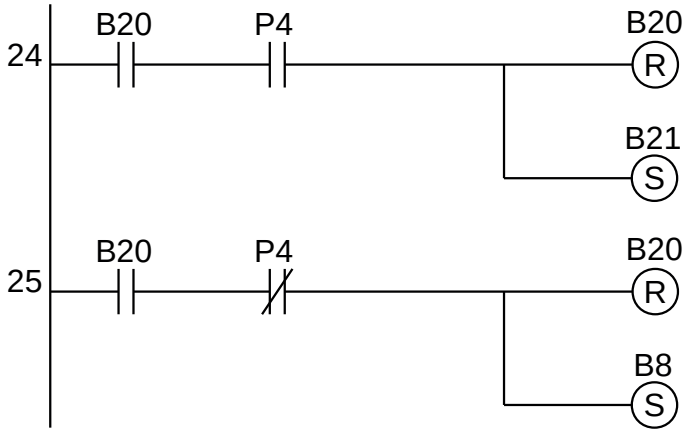

Figura 6.31 – Programa em linguagem Ladder para o bloco B20.

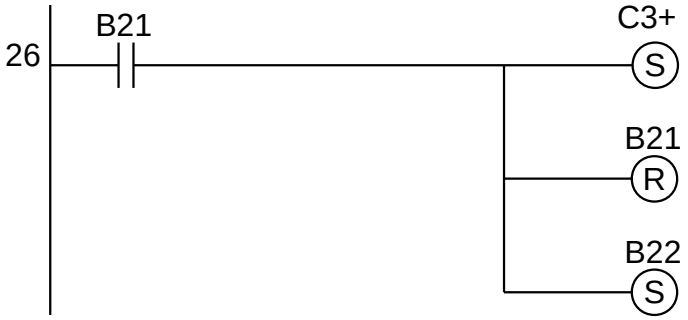

Figura 6.32 – Programa em linguagem Ladder para o bloco B21.

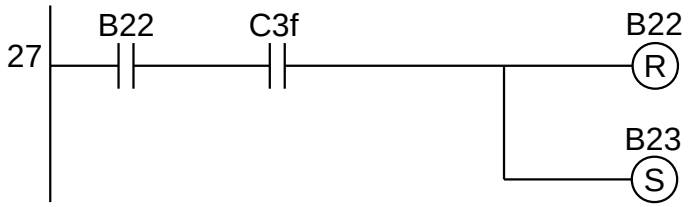

Figura 6.33 – Programa em linguagem Ladder para o bloco B22.

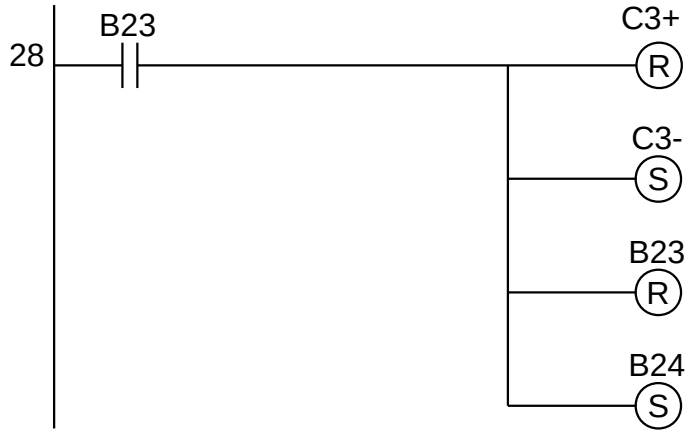

Figura 6.34 – Programa em linguagem Ladder para o bloco B23.

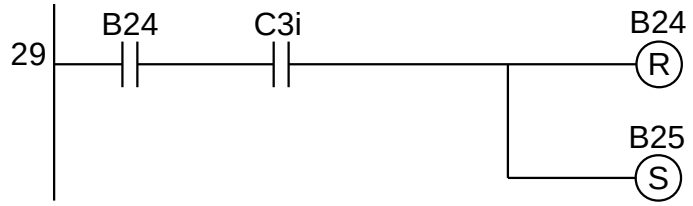

Figura 6.35 – Programa em linguagem Ladder para o bloco B24.

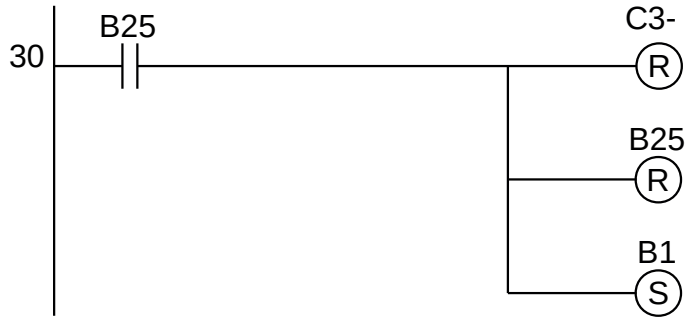

Figura 6.36 – Programa em linguagem Ladder para o bloco B25.

Figura 6.37 – Programa em linguagem Ladder para o bloco B26.

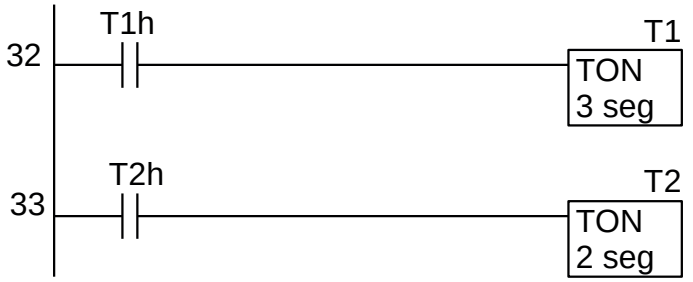

Figura 6.38 – Programa em linguagem Ladder para os temporizadores.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 6.6 e 6.7. As tabelas com as variáveis internas podem ser vistas nas tabelas 6.8.

Variável	Etiqueta	Descrição
M1	B1	Memória para o bloco 1.
M2	B2	Memória para o bloco 2.
M3	B3	Memória para o bloco 3.
M4	B4	Memória para o bloco 4.
M5	B5	Memória para o bloco 5
M6	B6	Memória para o bloco 6.
M7	B7	Memória para o bloco 7.
M8	B8	Memória para o bloco 8.
M9	B9	Memória para o bloco 9.
M10	B10	Memória para o bloco 10.
M11	B11	Memória para o bloco 11.
M12	B12	Memória para o bloco 12.
M13	B13	Memória para o bloco 13.
M14	B14	Memória para o bloco 14.
M15	B15	Memória para o bloco 15.
M16	B16	Memória para o bloco 16.
M17	B17	Memória para o bloco 17.
M18	B18	Memória para o bloco 18.
M19	B19	Memória para o bloco 19.
M20	B20	Memória para o bloco 20.
M21	B21	Memória para o bloco 21.
M22	B22	Memória para o bloco 22.
M23	B23	Memória para o bloco 23.
M24	B24	Memória para o bloco 24.
M25	B25	Memória para o bloco 25.
M26	Trava	Memória para emulação de "primeiro ciclo".
M27	T1h	Memória para habilitação de T1
M28	T2h	Memória para habilitação de T2
T01	T1	Temporizador para ligar (TON) 3 segundos.
T02	T2	Temporizador para ligar (TON) 2 segundos.

Tabela 6.8 – Descrição das variáveis internas.

6.7.3 – Exemplo 3. Máquina de encher frascos

Este exemplo mostra o uso de variáveis analógicas e a escolha múltipla de sequências.

Descrição da máquina

Uma máquina para encher frascos é mostrada na figura 6.39. Essa máquina permite encher frascos de 3 alturas diferentes com um líquido até uma determinada quantidade que depende do tamanho do frasco.

A máquina é constituída por 6 (seis) dispositivos:

1. Uma esteira de entrada acionada por um motor de indução que é energizado através de um contator com etiqueta “M1”.
2. Uma esteira de saída acionada por um motor de indução que é energizado através de um contator com etiqueta “M3”.
3. Uma esteira de pesagem acionada por um motor de indução que é energizado através de um contator com etiqueta “M2”.
4. Uma balança potenciométrica que fornece um sinal analógico de 0 a 10 V com etiqueta “B1”.
5. Um bico fornecedor de líquido acionado por uma válvula solenoide com etiqueta “V1”.
6. Um cilindro pneumático de dupla ação para descer e subir o bico de líquido até o frasco com uma eletroválvula controlada por dois solenoides para avanço com etiqueta A+ e para recuo com etiqueta A-.

Existem 3 sensores de proximidade do tipo capacitivo para identificar a presença de frascos nas esteiras, sendo o sensor S1 na esteira de entrada, o sensor S2 na esteira de pesagem e o sensor S3 na esteira de saída. Os sensores fornecem um sinal alto (verdadeiro) quando o frasco está defronte dele. Os sensores S1 e S2 são próximos de tal modo que um frasco possa sensibilizar os dois sensores.

O cilindro pneumático possui 4 sensores magnéticos do tipo “reed” para identificar o deslocamento do embolo do cilindro, sendo o sensor P0 para embolo totalmente recuado, o sensor P1 para embolo avançado até a borda do frasco alto, o sensor P2 para embolo avançado até a borda do frasco médio e o sensor P3 para embolo avançado até a borda do frasco pequeno.

Considere a existência de um painel de comando com uma chave rotativa para ligar e desligar o controlador, uma botoeira para iniciar o programa do controlador com a etiqueta “Partida” e uma botoeira para parar o programa do controlador com a etiqueta “Parada”. Existem uma lâmpada sinalizadora de defeitos com a etiqueta “Alarme”.

Considere como condições iniciais que o cilindro pneumático A está totalmente recuado, as esteiras desligadas e nenhum frasco sobre as esteiras de entrada, pesagem e saída.

O alarme deve ser ativado quando o valor do peso do frasco vazio estiver fora das especificações pedidas. Para este problema, o valor máximo de produto dentro do frasco não será considerado.

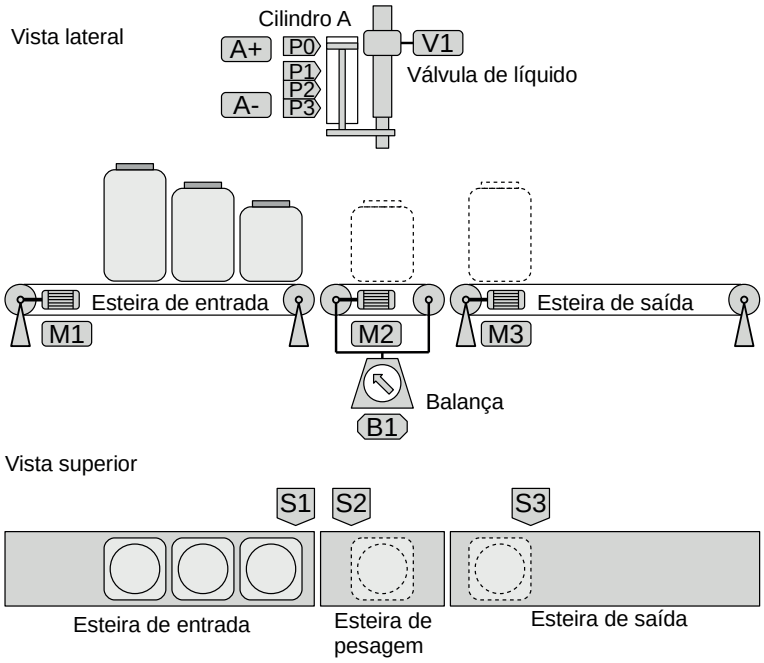

Figura 6.39 – Diagrama esquemático da máquina de encher frascos.

Descrição do funcionamento

Esteira de entrada

Ao iniciar o processo, a esteira de entrada é ligada (M1 = 1) até que o sensor S1 detecte o primeiro frasco, ou seja, o sensor S1 é verdadeiro (S1 = 1). Se a máquina estiver ocupada, a esteira de entrada deve ser desligada (M1 = 0). Se a máquina estiver livre, ou seja, não estiver enchendo um frasco, A esteira de entrada continua ligada (M1 = 1) e a esteira de pesagem é ligada (M2 = 1) até que o sensor S2 detecte o frasco, ou seja, o sensor S2 é verdadeiro (S2 = 1). A esteira de entrada é desligada (M1 = 0).

Esteira de pesagem

Se existir frasco na esteira de entrada e na posição de transferência, (S1 = 1), então a esteira de pesagem é ligada (M2 = 1) até que o frasco ative o sensor S2 e o faça verdadeiro (S2 = 1). A esteira de entrada é desligada (M1 = 0) e a esteira de pesagem continua ligada (M2 = 1), movimentando o frasco sobre a esteira, até que o frasco não sensibiliza mais o sensor S2 e este fica falso (S2 = 0), então a esteira de pesagem é desligada (M2 = 0).

Esteira de saída

Ao término do processo, a esteira de pesagem é ligada (M2 = 1) e a esteira de saída é ligada (M3 = 1) até que o frasco ative o sensor S3 e o faça verdadeiro (S3 = 1). A esteira de pesagem é desligada (M2 = 0) e a esteira de saída continua ligada (M3 = 1), movimentando o frasco sobre a esteira, até que o frasco não sensibiliza mais o sensor S3 e este fica falso (S3 = 0), então a esteira de saída é desligada (M3 = 0).

Avanço do cilindro A

Um sinal de comando de enchimento de frascos é feito verdadeiro e o embolo do cilindro A avança quando o solenoide A+ é feito verdadeiro (A+ = 1) até que um dos sensores Px (x = 1 ou 2 ou 3) seja verdadeiro (Px = 1), então o solenoide A+ é desligado (A+ = 0). A identificação do tamanho do frasco é realiza pela leitura do valor da balança potenciométrica (tabela 6.9).

Enchimento de produto

Quando o sensor Px for verdadeiro a eletroválvula V1 é aberta (V1 = 1) até que a quantidade determinada de líquido seja colocada dentro do frasco. A identificação da quantidade de líquido no frasco é realiza pela leitura do valor da balança potenciométrica (tabela 6.10).

Recuo do cilindro A

Um sinal de comando de finalização de enchimento de frascos é feito verdadeiro e o embolo do cilindro A recua quando o solenoide A- é feito verdadeiro (A- = 1) até que o sensor P0 seja verdadeiro (P0 = 1), então o solenoide A- é desligado (A- = 0).

Balança potenciométrica

A balança potenciométrica envia um sinal de tensão analógico para o controlador de acordo com as tabelas abaixo.

Frasco	Peso mínimo (V)	Peso médio (V)	Peso máximo (V)
Pequeno	0,70	1,00	1,30
Médio	1,65	2,00	2,35
Grande	2,60	3,00	3,40

Tabela 6.9 – Faixa de valores de tensão para o peso dos frascos vazios.

Frasco	Peso mínimo (V)	Peso máximo (V)
Pequeno	5,00	5,20
Médio	7,00	7,30
Grande	9,00	9,40

Tabela 6.10 – Faixa de valores de tensão para o peso dos frascos cheios.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entradas é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 6.11.

Parafuso	Etiqueta	Descrição
I01	Partida	Botoeira NA.
I02	Parada	Botoeira NA.
I03	S1	Sensor capacitivo de proximidade.
I04	S2	Sensor capacitivo de proximidade.
I05	S3	Sensor capacitivo de proximidade.
I06	P0	Sensor magnético de cilindro pneumático.
I07	P1	Sensor magnético de cilindro pneumático.
I08	P2	Sensor magnético de cilindro pneumático.
I09	P3	Sensor magnético de cilindro pneumático.
A01	B1	Balança potenciométrica.

Tabela 6.11 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 6.12.

Parafuso	Etiqueta	Descrição
Q01	M1	Contator de acionamento de motor de indução.
Q02	M2	Contator de acionamento de motor de indução.
Q03	M3	Contator de acionamento de motor de indução.
Q04	V1	Válvula solenoide de vazão.
Q05	A+	Válvula duplo solenoide pneumática 5/2 vias.
Q06	A-	Válvula duplo solenoide pneumática 5/2 vias.

Tabela 6.12 – Descrição das saídas.

Parte 1 – Inicialização. Solução por frases lógicas

É criado a variável “Func” que indica que o programa do controlador pode ser executado.

Esta variável é controlada pelas botoeiras “Partida” e “Parada”. Também é criada a variável “Trava” para inibir o reinício do primeiro bloco do fluxograma através da botoeira de inicialização.

É utilizado frases lógicas para a modelagem dessa etapa.

A primeira frase produz um “pulso único” da botoeira “Partida”. Esse pulso ativa o primeiro bloco do fluxograma.

SE (“Partida” é verdadeiro e “Trava” é falso) ENTÃO (fazer “B01” verdadeiro e memorizar e fazer “Trava” verdadeiro e memorizar)

A segunda frase permite o reinício do programa, habilitando o funcionamento do programa.

SE (“Partida” é verdadeiro) ENTÃO (fazer a variável “Func” verdadeiro e memorizar).

A terceira frase desabilita a variável de funcionamento do programa.

SE (“Parada” é verdadeiro) ENTÃO (fazer a variável “Func” falso).

Parte 2 – Funcionamento. Solução por fluxograma

O fluxograma de funcionamento normal desta máquina pode ser visto nas figuras 6.40 e 6.41.

Condição inicial

O cilindro pneumático A está totalmente recuado, as esteiras estão desligadas e nenhum frasco está sobre as esteiras de entrada, pesagem e saída.

Posicionamento do frasco

Bloco B01 (decisão)

O controlador fica esperando se a variável “Func” é verdadeira, habilitando o funcionamento do programa. A lógica booleana usada é “Func”. Se a lógica for verdadeira, o bloco B01 é desativado e o bloco B02 é ativado. Se a lógica for falsa, o bloco B01 não é desativado.

Bloco B02 (ação)

A esteira de entrada é ligada. A saída M1 é ativada, o bloco B02 é desativado e o bloco B03 é ativado.

Bloco B03 (decisão)

O controlador fica esperando um frasco chegar ao final da esteira de entrada, ou seja, se a variável “S1” é verdadeira. A lógica booleana usada é “S1”. Se a lógica for verdadeira, o bloco B03 é desativado e o bloco B04 é ativado. Se a lógica for falsa, o bloco B03 não é desativado.

Bloco B04 (ação)

A esteira de pesagem é ligada. A saída M2 é ativada, o bloco B04 é desativado e o bloco B05 é ativado.

Bloco B05 (decisão)

O frasco é transferido da esteira de entrada para a esteira de pesagem, ou seja, se a variável “S2” é verdadeira. A lógica booleana usada é “S2”. Se a lógica for verdadeira, o bloco B05 é desativado e o bloco B06 é ativado. Se a lógica for falsa, o bloco B05 não é desativado.

Bloco B06 (ação)

A esteira de entrada é desligada. A saída M1 é desativada, o bloco B06 é desativado e o bloco B07 é ativado.

Bloco B07 (decisão)

O frasco é posicionado na esteira de pesagem, ou seja, se a variável "S2" é falsa. A lógica booleana usada é "$\overline{S2}$". Se a lógica for verdadeira, o bloco B07 é desativado e o bloco B08 é ativado. Se a lógica for falsa, o bloco B07 não é desativado.

Bloco B08 (ação)

A esteira de pesagem é desligada. A saída M2 é desativada, o bloco B08 é desativado e o bloco B09 é ativado.

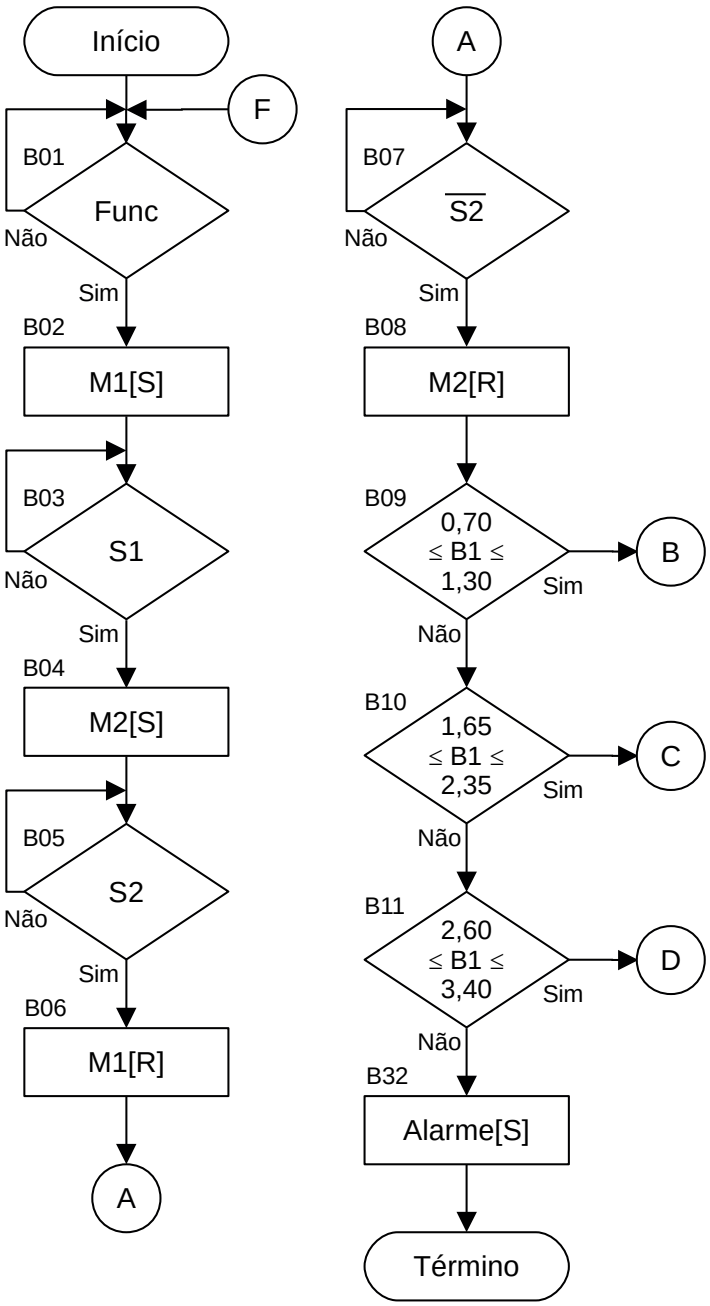

Figura 6.40 – Fluxograma operacional para a máquina de enchimento de frascos (parte 1).

Identificação do tamanho do frasco

Bloco B09 (decisão).

O frasco é testado para o tamanho pequeno, ou seja, se o peso do frasco está entre os valores de 0,7 e 1,3. Um comparador de janela (Comp1) é utilizado, resultando em saída verdadeira se o valor de entrada (B1) está entre os limites inferior e superior. A lógica booleana usada é “Comp1”. Se a lógica for verdadeira, o bloco B09 é desativado e o bloco B12 é ativado. Se a lógica for falsa, o bloco B09 é desativado e o bloco B10 é ativado.

Bloco B10 (decisão).

O frasco é testado para o tamanho médio, ou seja, se o peso do frasco está entre os valores de 1,65 e 2,35. Um comparador de janela (Comp2) é utilizado, resultando em saída verdadeira se o valor de entrada (B1) está entre os limites inferior e superior. A lógica booleana usada é “Comp2”. Se a lógica for verdadeira, o bloco B10 é desativado e o bloco B16 é ativado. Se a lógica for falsa, o bloco B10 é desativado e o bloco B11 é ativado.

Bloco B11 (decisão).

O frasco é testado para o tamanho grande, ou seja, se o peso do frasco está entre os valores de 2,60 e 3,40. Um comparador de janela (Comp3) é utilizado, resultando em saída verdadeira se o valor de entrada (B1) está entre os limites inferior e superior. A lógica booleana usada é “Comp3”. Se a lógica for verdadeira, o bloco B11 é desativado e o bloco B20 é ativado. Se a lógica for falsa, o bloco B11 é desativado.

Bloco B32 (ação).

É feito a sinalização de alarme. A saída Alarme é ativada, o bloco B32 é desativado.

Avanço do cilindro pneumático e colocação de líquido no frasco pequeno

Bloco B12 (ação).

A eletroválvula de avanço do cilindro pneumático “A” é aberta e o embolo avança. A saída A+ é desativada, o bloco B12 é desativado e o bloco B13 é ativado.

Bloco B13 (decisão).

O embolo do cilindro pneumático “A” avança até a posição do frasco pequeno, ou seja, o sensor “P3” é ativado. A lógica booleana usada é “P3”. Se a lógica for verdadeira, o bloco B13 é desativado e o bloco B14 é ativado. Se a lógica for falsa, o bloco B13 não é desativado.

Bloco B14 (ação).

A eletroválvula de avanço do cilindro pneumático “A” é fechada e o embolo para o avanço. A eletroválvula “V1” é aberta e o líquido começa a cair no frasco pequeno. A saída A+ é desativada, a saída “V1” é ativada, o bloco B14 é desativado e o bloco B15 é ativado.

Bloco B15 (decisão).

O frasco com o líquido é testado para o peso de líquido no frasco pequeno, ou seja, se o peso do frasco com o líquido é maior que 5,00. Um comparador de magnitude (Comp4) é utilizado, resultando em saída verdadeira se o valor de entrada (B1) é maior que o limite inferior. A lógica booleana usada é “Comp4”. Se a lógica for verdadeira, o bloco B15 é desativado e o bloco B24 é ativado. Se a lógica for falsa, o bloco B15 não é desativado.

Avanço do cilindro pneumático e colocação de líquido no frasco médio

Bloco B16 (ação).

A eletroválvula de avanço do cilindro pneumático "A" é aberta e o embolo avança. A saída A+ é desativada, o bloco B16 é desativado e o bloco B17 é ativado.

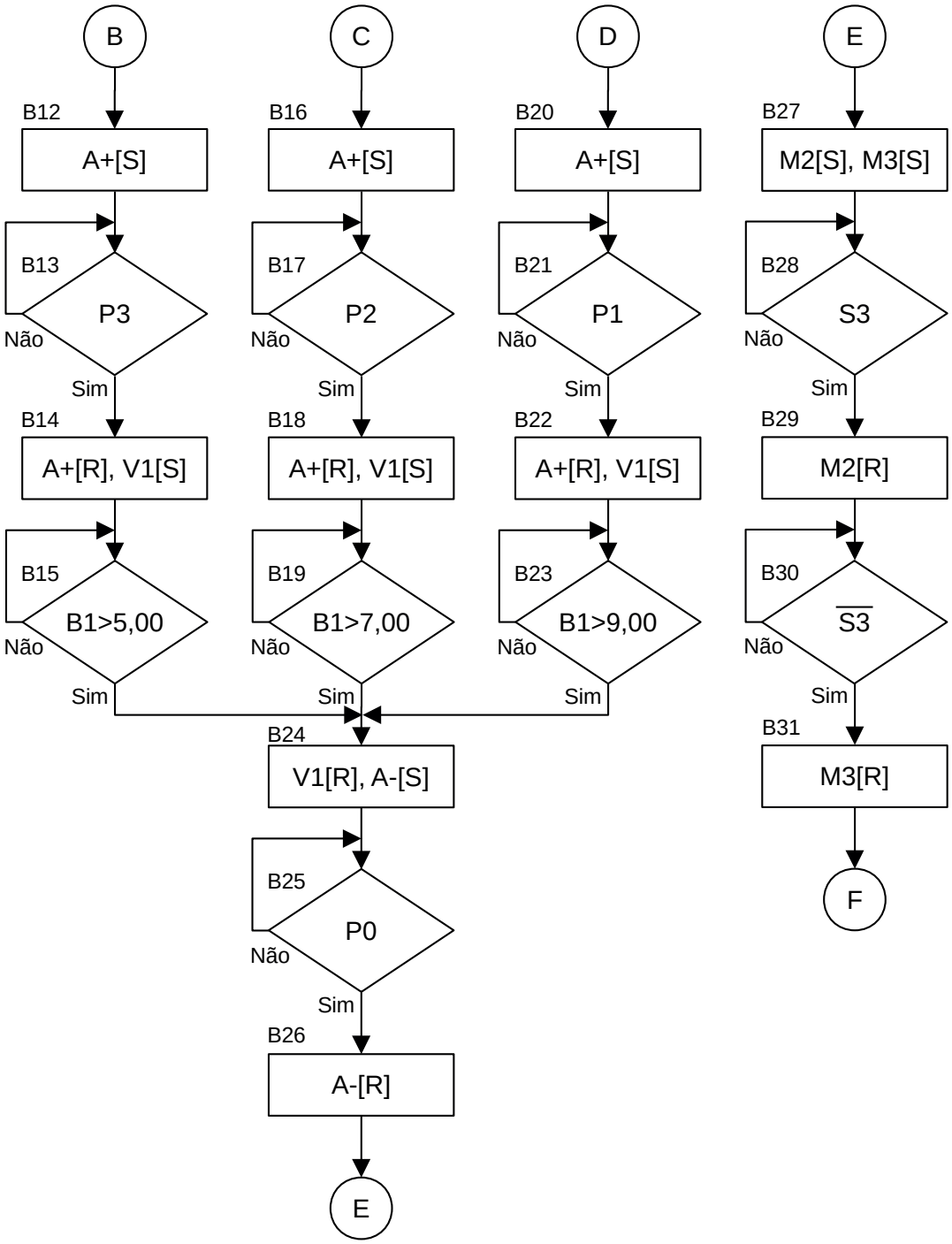

Figura 6.41 – Fluxograma operacional para a máquina de enchimento de frascos (parte 2).

Bloco B17 (decisão)

O embolo do cilindro pneumático "A" avança até a posição do frasco médio, ou seja, o sensor "P2" é ativado. A lógica booleana usada é "P2". Se a lógica for verdadeira, o bloco B17 é desativado e o bloco B18 é ativado. Se a lógica for falsa, o bloco B17 não é desativado.

Bloco B18 (ação)

A eletroválvula de avanço do cilindro pneumático “A” é fechada e o embolo para o avanço. A eletroválvula “V1” é aberta e o líquido começa a cair no frasco médio. A saída A+ é desativada, a saída “V1” é ativada, o bloco B18 é desativado e o bloco B19 é ativado.

Bloco B19 (decisão)

O frasco com o líquido é testado para o peso de líquido no frasco médio, ou seja, se o peso do frasco com o líquido é maior que 7,00. Um comparador de magnitude (Comp5) é utilizado, resultando em saída verdadeira se o valor de entrada (B1) é maior que o limite inferior. A lógica booleana usada é “Comp5”. Se a lógica for verdadeira, o bloco B19 é desativado e o bloco B24 é ativado. Se a lógica for falsa, o bloco B19 não é desativado.

Avanço do cilindro pneumático e colocação de líquido no frasco grande

Bloco B20 (ação)

A eletroválvula de avanço do cilindro pneumático “A” é aberta e o embolo avança. A saída A+ é desativada, o bloco B20 é desativado e o bloco B21 é ativado.

Bloco B21 (decisão)

O embolo do cilindro pneumático “A” avança até a posição do frasco grande, ou seja, o sensor “P1” é ativado. A lógica booleana usada é “P1”. Se a lógica for verdadeira, o bloco B21 é desativado e o bloco B22 é ativado. Se a lógica for falsa, o bloco B21 não é desativado.

Bloco B22 (ação)

A eletroválvula de avanço do cilindro pneumático “A” é fechada e o embolo para o avanço. A eletroválvula “V1” é aberta e o líquido começa a cair no frasco grande. A saída A+ é desativada, a saída “V1” é ativada, o bloco B22 é desativado e o bloco B23 é ativado.

Bloco B23 (decisão)

O frasco com o líquido é testado para o peso de líquido no frasco grande, ou seja, se o peso do frasco com o líquido é maior que 9,00. Um comparador de magnitude (Comp6) é utilizado, resultando em saída verdadeira se o valor de entrada (B1) é maior que o limite inferior. A lógica booleana usada é “Comp6”. Se a lógica for verdadeira, o bloco B23 é desativado e o bloco B24 é ativado. Se a lógica for falsa, o bloco B23 não é desativado.

Fechamento da válvula de líquido e recuo do cilindro pneumático

Bloco B24 (ação)

A eletroválvula de recuo do cilindro pneumático “A” é aberta e o embolo inicia o recuo. A eletroválvula “V1” é fechada e o líquido deixa de cair no frasco. A saída A- é ativada, a saída “V1” é desativada, o bloco B24 é desativado e o bloco B25 é ativado.

Bloco B25 (decisão)

O embolo do cilindro pneumático “A” recua até a posição de início, ou seja, o sensor “P0” é ativado. A lógica booleana usada é “P0”. Se a lógica for verdadeira, o bloco B25 é desativado e o bloco B26 é ativado. Se a lógica for falsa, o bloco B25 não é desativado.

Bloco B26 (ação)

A eletroválvula de recuo do cilindro pneumático “A” é fechada e o embolo para o recuo. A saída A- é desativada, o bloco B26 é desativado e o bloco B27 é ativado.

Transferência do frasco cheio para a esteira de saída

Bloco B27 (ação)

As esteiras de pesagem e saída são ligadas para transferir o frasco cheio. A saída M2 é ativada, a saída M3 é ativada, o bloco B27 é desativado e o bloco B28 é ativado.

Nota: Observe que o conteúdo do bloco B27 pode ficar junto ao bloco B26. As ações foram separadas em dos blocos apenas para separação dos eventos.

Bloco B28 (decisão)

O frasco é transferido da esteira de pesagem para a esteira de saída, ou seja, se a variável "S3" é verdadeira. A lógica booleana usada é "S3". Se a lógica for verdadeira, o bloco B28 é desativado e o bloco B29 é ativado. Se a lógica for falsa, o bloco B28 não é desativado.

Bloco B29 (ação)

A esteira de pesagem é desligada. A saída M2 é desativada, o bloco B29 é desativado e o bloco B30 é ativado.

Bloco B30 (decisão)

O frasco é posicionado na esteira de saída, ou seja, se a variável "S3" é falsa. A lógica booleana usada é "$\overline{S3}$". Se a lógica for verdadeira, o bloco B30 é desativado e o bloco B31 é ativado. Se a lógica for falsa, o bloco B30 não é desativado.

Bloco B31 (ação)

A esteira de saída é desligada. A saída M3 é desativada, o bloco B31 é desativado e o bloco B01 é ativado.

Codificação do fluxograma em linguagem Ladder

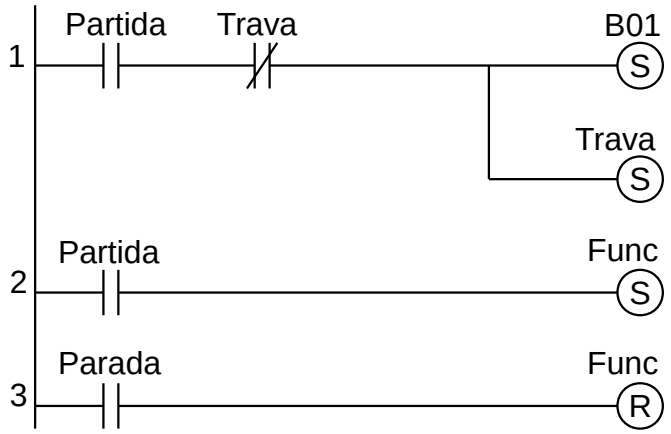

Figura 6.42 – Programa em linguagem Ladder para a inicialização.

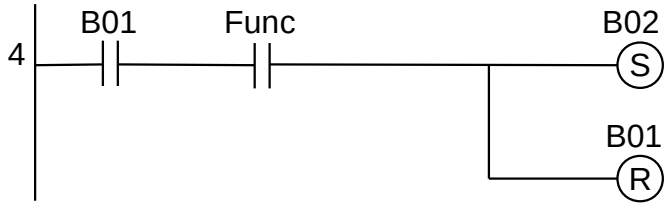

Figura 6.43 – Programa em linguagem Ladder para o bloco B01.

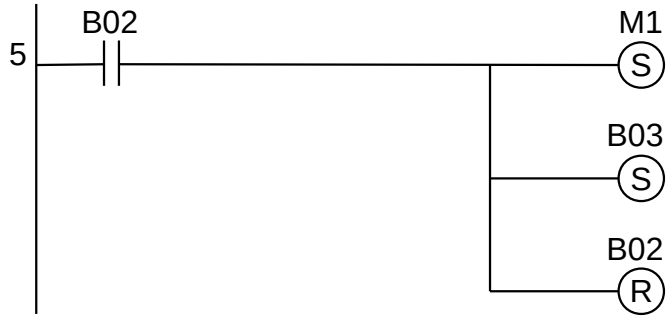

Figura 6.44 – Programa em linguagem Ladder para o bloco B02.

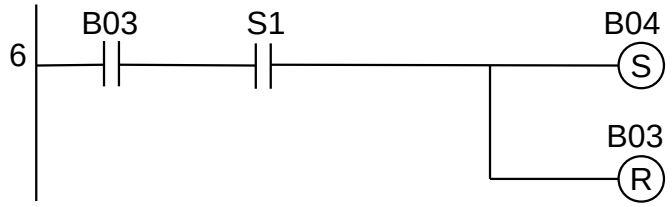

Figura 6.45 – Programa em linguagem Ladder para o bloco B03.

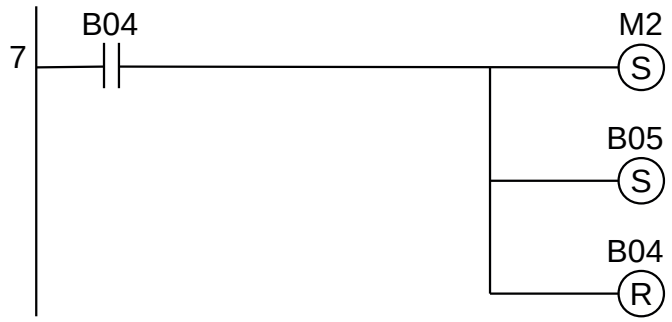

Figura 6.46 – Programa em linguagem Ladder para o bloco B04.

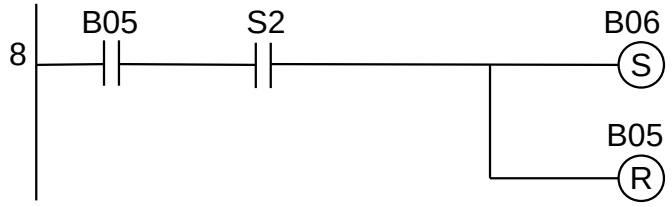

Figura 6.47 – Programa em linguagem Ladder para o bloco B05.

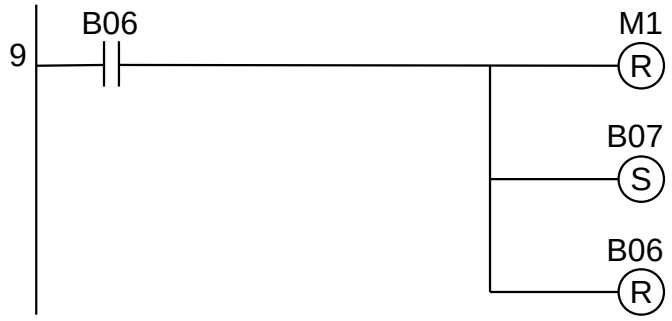

Figura 6.48 – Programa em linguagem Ladder para o bloco B06.

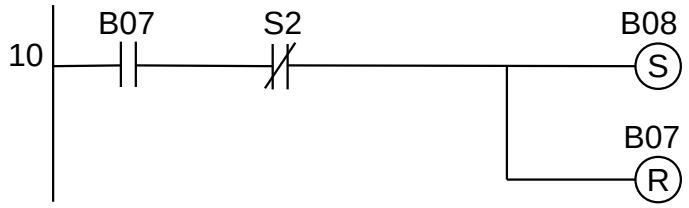

Figura 6.49 – Programa em linguagem Ladder para o bloco B07.

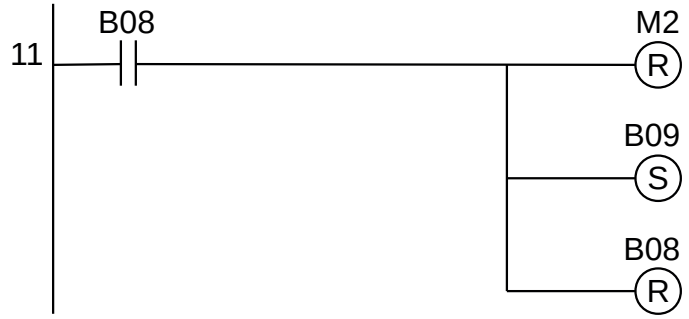

Figura 6.50 – Programa em linguagem Ladder para o bloco B08.

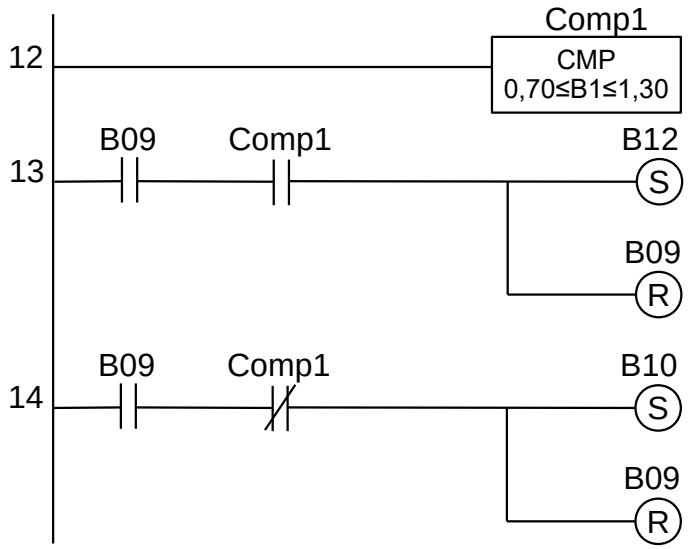

Figura 6.51 – Programa em linguagem Ladder para o bloco B09.

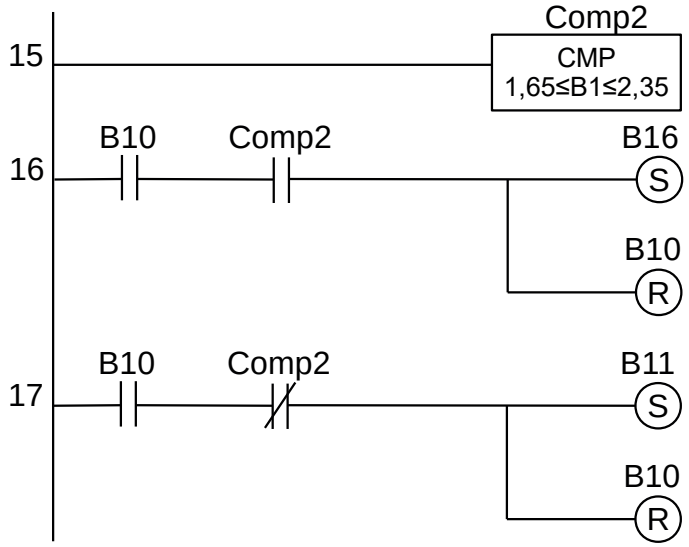

Figura 6.52 – Programa em linguagem Ladder para o bloco B10.

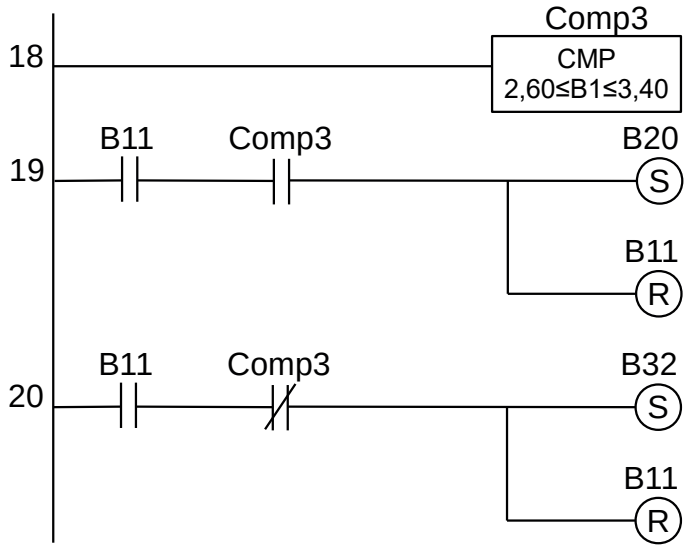

Figura 6.53 – Programa em linguagem Ladder para o bloco B11.

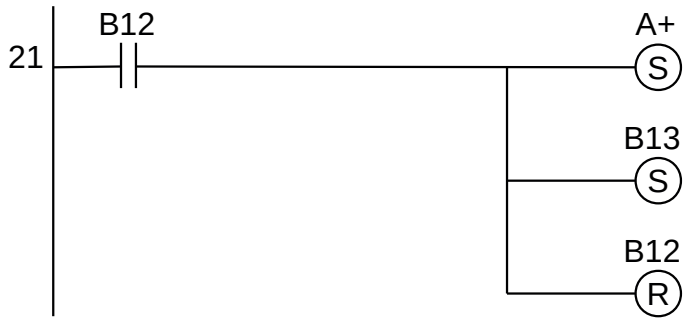

Figura 6.54 – Programa em linguagem Ladder para o bloco B12.

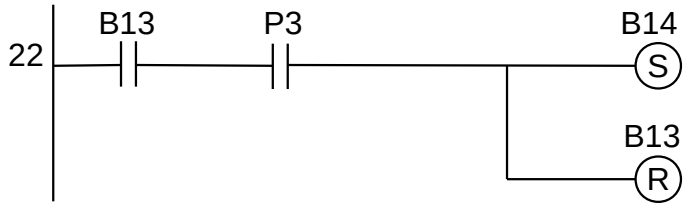

Figura 6.55 – Programa em linguagem Ladder para o bloco B13.

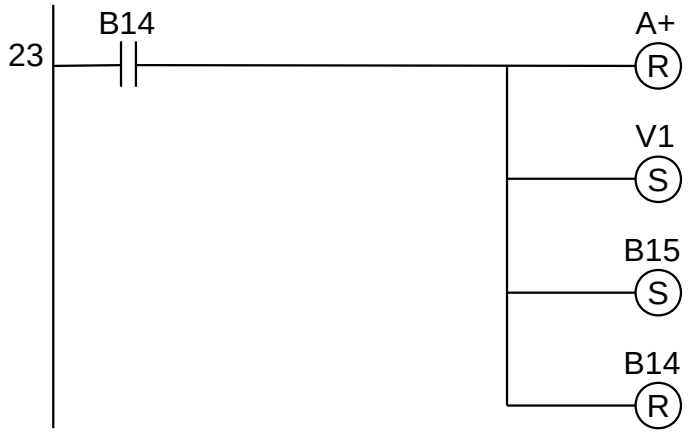

Figura 6.56 – Programa em linguagem Ladder para o bloco B14.

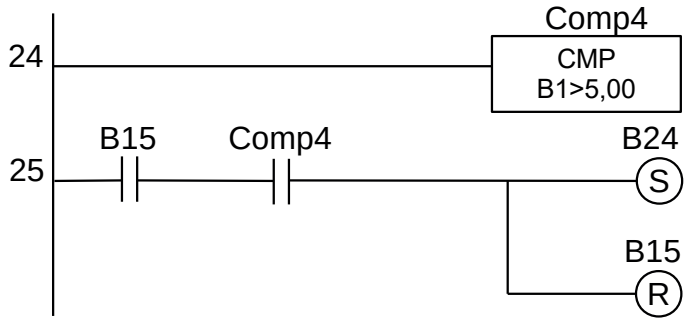

Figura 6.57 – Programa em linguagem Ladder para o bloco B15.

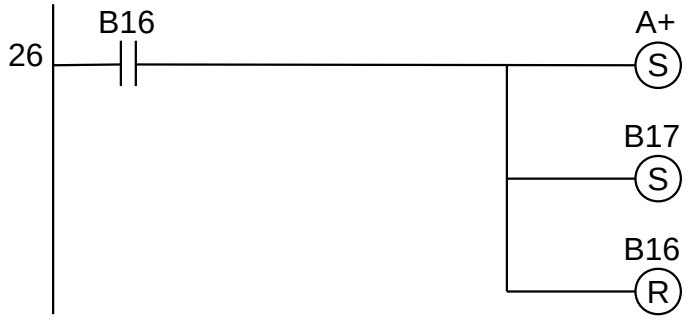

Figura 6.58 – Programa em linguagem Ladder para o bloco B16.

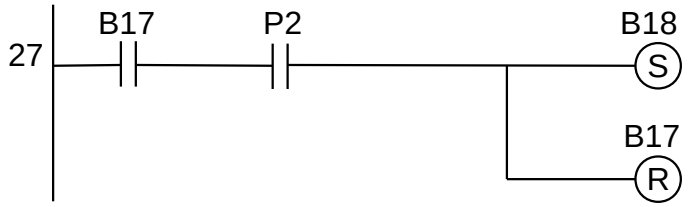

Figura 6.59 – Programa em linguagem Ladder para o bloco B17.

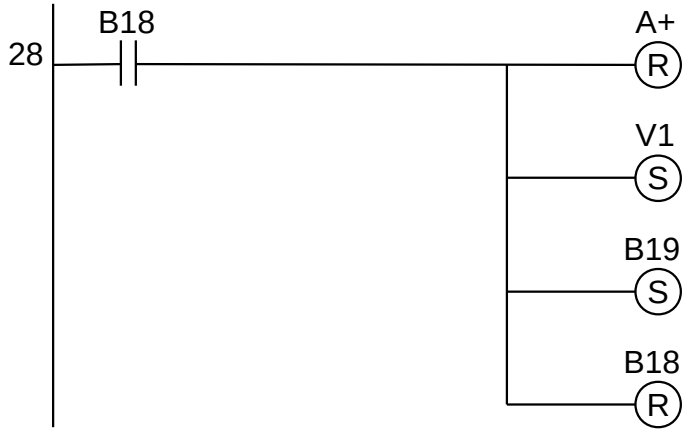

Figura 6.60 – Programa em linguagem Ladder para o bloco B18.

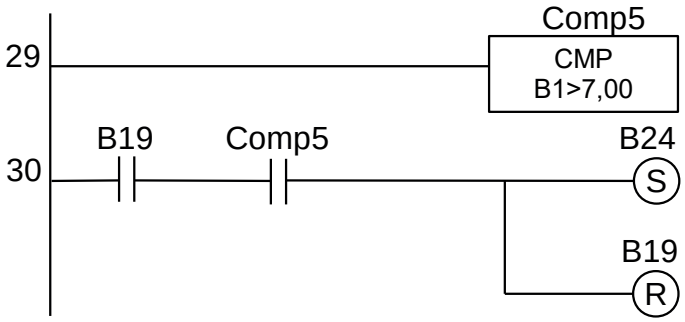

Figura 6.61 – Programa em linguagem Ladder para o bloco B19.

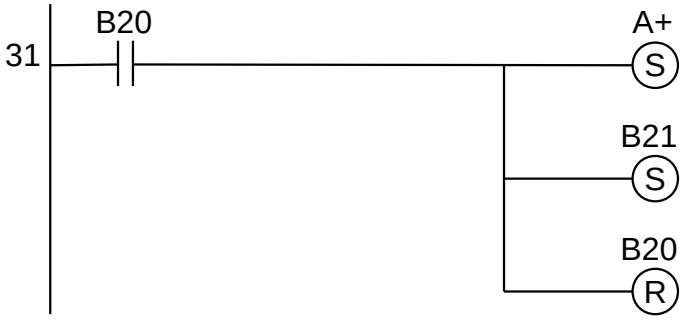

Figura 6.62 – Programa em linguagem Ladder para o bloco B20.

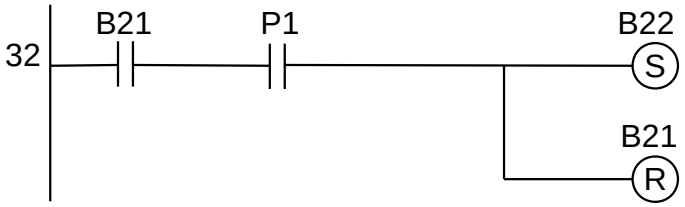

Figura 6.63 – Programa em linguagem Ladder para o bloco B21.

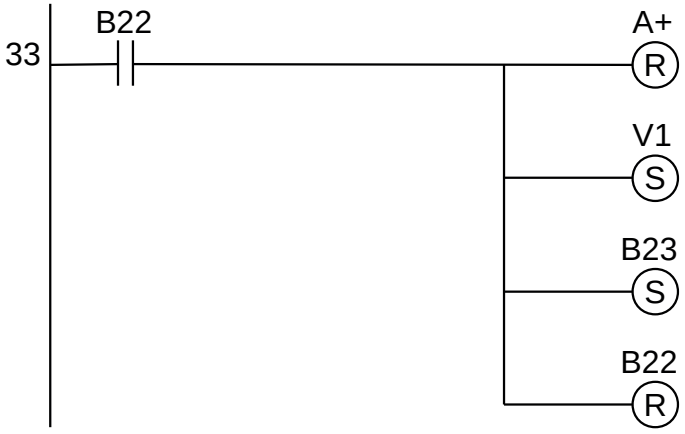

Figura 6.64 – Programa em linguagem Ladder para o bloco B22.

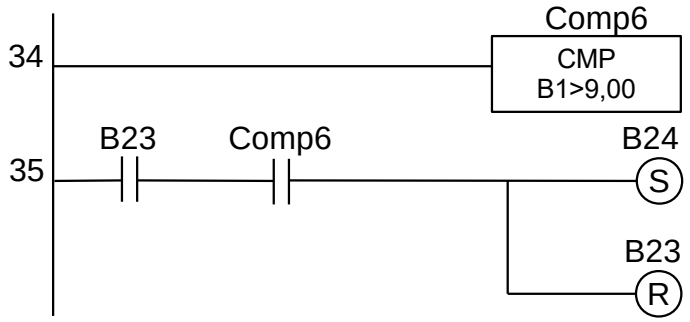

Figura 6.65 – Programa em linguagem Ladder para o bloco B23.

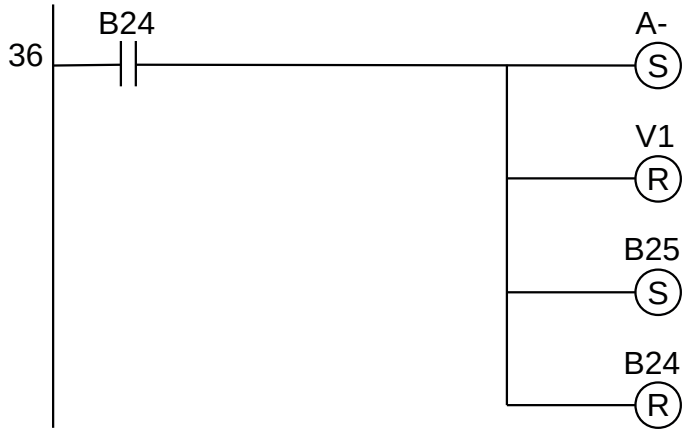

Figura 6.66 – Programa em linguagem Ladder para o bloco B24.

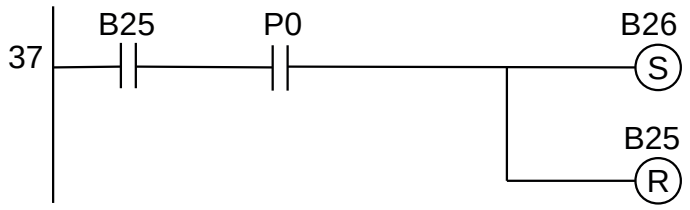

Figura 6.67 – Programa em linguagem Ladder para o bloco B25.

Figura 6.68 – Programa em linguagem Ladder para o bloco B26.

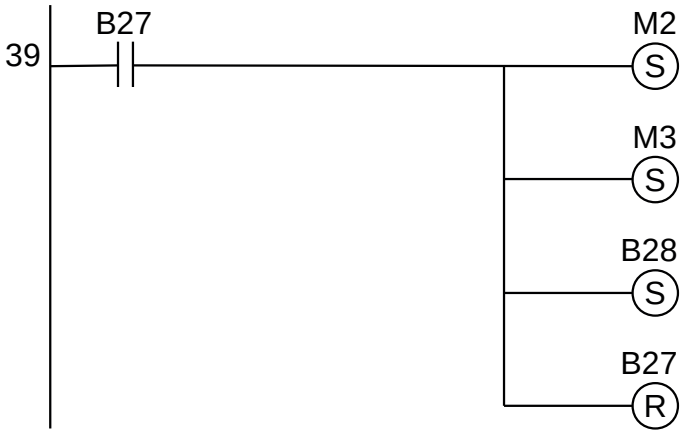

Figura 6.69 – Programa em linguagem Ladder para o bloco B27.

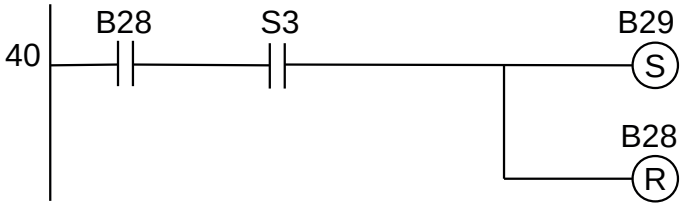

Figura 6.70 – Programa em linguagem Ladder para o bloco B28.

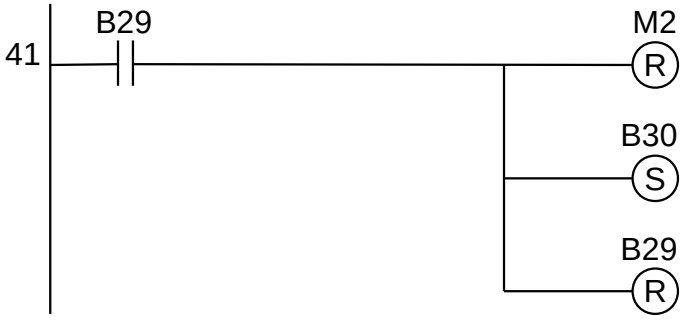

Figura 6.71 – Programa em linguagem Ladder para o bloco B29.

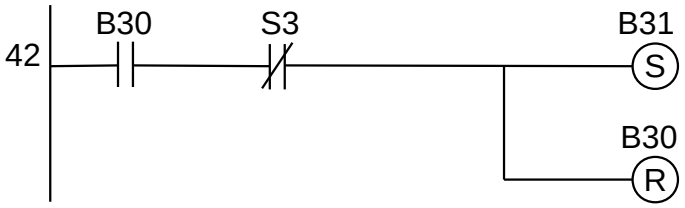

Figura 6.72 – Programa em linguagem Ladder para o bloco B30.

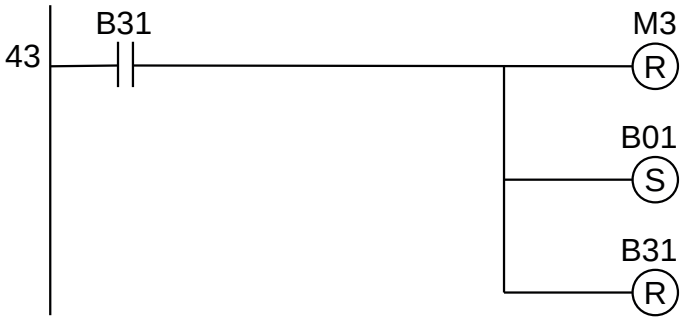

Figura 6.73 – Programa em linguagem Ladder para o bloco B31.

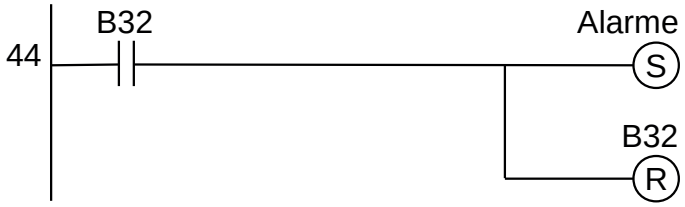

Figura 6.74 – Programa em linguagem Ladder para o bloco B32.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 6.11 e 6.12. As tabelas com as variáveis internas podem ser vistas na tabela 6.13.

Variável	Etiqueta	Descrição
M01	B01	Memória para o bloco B01.
M02	B02	Memória para o bloco B02.
M03	B03	Memória para o bloco B03.
M04	B04	Memória para o bloco B04.
M05	B05	Memória para o bloco B05.
M06	B06	Memória para o bloco B06.
M07	B07	Memória para o bloco B07.
M08	B08	Memória para o bloco B08.
M09	B09	Memória para o bloco B09.
M10	B10	Memória para o bloco B10.
M11	B11	Memória para o bloco B11.
M12	B12	Memória para o bloco B12.
M13	B13	Memória para o bloco B13.
M14	B14	Memória para o bloco B14.
M15	B15	Memória para o bloco B15.
M16	B16	Memória para o bloco B16.
M17	B17	Memória para o bloco B17.
M18	B18	Memória para o bloco B18.
M19	B19	Memória para o bloco B19.
M20	B20	Memória para o bloco B20.
M21	B21	Memória para o bloco B21.
M22	B22	Memória para o bloco B22.
M23	B23	Memória para o bloco B23.
M24	B24	Memória para o bloco B24.
M25	B25	Memória para o bloco B25.
M26	B26	Memória para o bloco B26.
M27	B27	Memória para o bloco B27.

Tabela 6.13 – Descrição das variáveis internas (parte 1).

Variável	Etiqueta	Descrição
M28	B28	Memória para o bloco B28.
M29	B29	Memória para o bloco B29.
M30	B30	Memória para o bloco B30.
M31	B31	Memória para o bloco B31.
M32	Func	Memória de funcionamento do programa.
M33	Trava	Memória de trava de inicialização.
CP1	Comp1	Comparador.
CP2	Comp2	Comparador.
CP3	Comp3	Comparador.
CP4	Comp4	Comparador.
CP5	Comp5	Comparador.
CP6	Comp6	Comparador.

Tabela 6.13 – Descrição das variáveis internas (parte 2).

6.8 – Conclusão

O fluxograma foi a primeira estrutura gráfica utilizada na elaboração de programas para computador. Para cada tipo de equipamento e linguagem foi elaborado novas formas geométricas para representar os diversos comandos e ações da máquina. Muitos desses símbolos relacionados a equipamento periférico do computador, não são mais usados devido a que os tipos de equipamentos periféricos foram deixados de usar. Por exemplo: fita magnética.

Com a evolução dos equipamentos periféricos e dos comandos das linguagens, novos símbolos são agregados à estrutura do fluxograma. Entretanto, novas estruturas gráficas foram criadas para facilitar a programação nas diversas linguagens e com as exigências dos novos equipamentos.

Para a programação de CLP, o fluxograma apresenta algumas desvantagens em relação às novas estruturas gráficas, como eventos simultâneos, condições de espera, variações de parâmetros e outros.

Capítulo 7

Estrutura de programação por diagrama de estados

7.1 – Introdução

Uma máquina de estado é qualquer dispositivo que armazena o status de um objeto em um determinado momento e pode alterar o status ou causar outras ações com base na entrada que recebe. Os estados referem-se às diferentes combinações de informações que um objeto pode conter, não como o objeto se comporta. Para entender os diferentes estados de um objeto, pode-se desejar visualizar todos os estados possíveis e mostrar como um objeto chega a cada estado.

Cada diagrama de estado normalmente começa com um círculo escuro que indica o estado inicial e termina com um círculo delimitado que indica o estado final. No entanto, apesar de ter pontos iniciais e finais claros, os diagramas de estado não são necessariamente a melhor ferramenta para capturar uma progressão geral de eventos. Em vez disso, eles ilustram tipos específicos de comportamento, em particular, mudanças de um estado para outro.

Os diagramas de estado representam principalmente estados e transições. Os estados são representados por retângulos com cantos arredondados identificados com o nome do estado. As transições são marcadas com setas que fluem de um estado para outro, mostrando como os estados mudam.

O diagrama de estados tende a ser uma forma eficiente de modelar as interações e colaborações nas entidades externas e no sistema. Ele modela sistemas baseados em eventos para lidar com o estado de um objeto. Ele também define vários estados distintos de um componente dentro do sistema. Cada objeto ou componente possui um estado específico.

Os tipos de diagrama de estado são:

Máquina de estado comportamental

O diagrama de máquina de estado comportamental registra o comportamento de um objeto dentro do sistema. Ele descreve a implementação de uma entidade específica. Ele modela o comportamento do sistema.

Máquina de estado de protocolo

Ele captura o comportamento do protocolo. A máquina de estado do protocolo descreve a mudança no estado do protocolo e mudanças paralelas dentro do sistema. Mas não retrata a implementação de um componente específico.

Usos do diagrama de estados

Um diagrama de estado é usado para representar a condição do sistema ou parte do sistema em instâncias finitas de tempo. É um diagrama comportamental e representa o comportamento usando transições de estado finito. Os diagramas de estado também são chamados de máquinas de estado e diagramas de gráfico de estado. Esses termos são frequentemente usados de forma intercambiável.

De forma simples, um diagrama de estado é usado para modelar o comportamento dinâmico de uma classe em resposta ao tempo e à mudança de estímulos externos. Pode-se dizer que cada classe tem um estado, mas não modelamos cada classe usando diagramas de estado.

Os principais usos são para declarar os eventos responsáveis pela mudança de estado, modelar o comportamento dinâmico do sistema e para compreender a reação de objetos ou classes a estímulos internos ou externos.

Os diagramas de estado são usados para implementar modelos de trabalho da vida real e sistemas orientados a objetos em profundidade. Esses diagramas são usados para comparar a natureza dinâmica e estática de um sistema, capturando o comportamento dinâmico de um sistema.

Os diagramas de estado são usados para capturar as mudanças em várias entidades do sistema do início ao fim. Eles são usados para analisar como um evento pode desencadear mudanças em vários estados de um sistema.

7.2 – Técnicas comuns de modelagem

As máquinas de estado são usadas mais comumente para modelar o comportamento de um objeto ao longo de seu tempo de vida. Eles são particularmente necessários quando os objetos têm comportamento dependente do estado. Os objetos que podem ter máquinas de estado incluem classes, subsistemas, casos de uso e interfaces (para afirmar estados que devem ser satisfeitos por um objeto que realiza a interface).

No caso de sistemas de tempo real, as máquinas de estado também são usadas para cápsulas e protocolos (para afirmar estados que devem ser satisfeitos por um objeto que realiza o protocolo).

Nem todos os objetos requerem máquinas de estado. Se o comportamento de um objeto é simples, de forma que ele simplesmente armazena ou recupera dados, o comportamento do objeto é invariante de estado e sua máquina de estado é de pouco interesse.

Modelar o tempo de vida de um objeto envolve três coisas: especificar os eventos aos quais o objeto pode responder, a resposta a esses eventos e o impacto do passado no comportamento atual. Modelar o tempo de vida de um objeto também envolve decidir a ordem em que o objeto pode responder significativamente aos eventos, começando no momento da criação do objeto e continuando até sua destruição.

Para modelar a vida útil de um objeto:

1. Defina o contexto para a máquina de estado, seja uma classe, um caso de uso ou o sistema como um todo.
   a) Se o contexto for uma classe ou um caso de uso, colete as classes vizinhas, incluindo classes pai ou classes acessíveis por associações ou dependências. Esses vizinhos são alvos candidatos para ações e são alvos candidatos para inclusão nas condições de guarda.
   b) Se o contexto for o sistema como um todo, restrinja seu foco a um comportamento do sistema e, a seguir, considere o tempo de vida dos objetos envolvidos naquele aspecto. A vida útil de todo o sistema é simplesmente muito grande para ser um foco significativo.
2. Estabeleça os estados inicial e final para o objeto. Se houver pré-condições ou pós-condições dos estados inicial e final, defina-as também.
3. Determine os eventos aos quais o objeto responde. Eles podem ser encontrados nas interfaces do objeto. No caso de sistemas de tempo real, eles podem ser encontrados nos protocolos do objeto.

4. Partindo do estado inicial para o estado final, defina os estados de nível superior em que o objeto pode estar. Conecte esses estados com as transições acionadas pelos eventos apropriados. Continue adicionando essas transições.
5. Identifique quaisquer ações de entrada ou saída.
6. Expanda ou simplifique a máquina de estado usando subestados.
7. Verifique se todos os eventos que acionam as transições na máquina de estado correspondem aos eventos esperados pelas interfaces realizadas pelo objeto. Da mesma forma, verifique se todos os eventos esperados pelas interfaces do objeto são tratados pela máquina de estado. No caso de sistemas de tempo real, faça verificações equivalentes para os protocolos de uma cápsula. Por fim, procure lugares onde você deseja explicitamente ignorar eventos (por exemplo, eventos adiados).
8. Verifique se todas as ações na máquina de estado são suportadas por relacionamentos, métodos e operações do objeto envolvente.
9. Rastreie a máquina de estado, comparando-a com as sequências esperadas de eventos e suas respostas. Pesquise estados inacessíveis e estados nos quais a máquina fique presa.
10. Se você reorganizar ou reestruturar a máquina de estado, verifique se a semântica não mudou.

Dicas

Quando puder escolher, use a semântica visual da máquina de estado em vez de escrever o código de transição de detalhes. Por exemplo, não acione uma transição em vários sinais e, em seguida, use o código de detalhe para gerenciar o fluxo de controle de forma diferente, dependendo do sinal. Use transições separadas, acionadas por sinais separados. Evite lógica condicional no código de transição que oculta comportamento adicional.

Nomeie os estados de acordo com o que você está esperando ou o que está acontecendo durante o estado. Lembre-se de que um estado não é um "ponto no tempo"; é um período durante o qual a máquina de estado espera que algo aconteça. Por exemplo, "EsperaraParaFim" é um nome melhor do que "fim"'; "EsperandoAlgumaAtividade" é melhor do que "TempoParado". Não nomeie os estados como se fossem ações.

Nomeie todos os estados e transições em uma máquina de estado de maneira exclusiva; isso tornará a depuração no nível da fonte mais fácil.

Use variáveis de estado (atributos usados para controlar o comportamento) com cautela; não os use em vez de criar novos estados. Onde os estados são poucos, com pouco ou nenhum comportamento dependente do estado e onde há pouco ou nenhum comportamento que pode ser concorrente ou independente do objeto que contém a máquina de estado, as variáveis de estado podem ser usadas.

Se houver um comportamento dependente de estado complexo que seja potencialmente simultâneo, ou se os eventos que devem ser manipulados puderem se originar fora do objeto que contém a máquina de estado, considere o uso de uma colaboração de dois ou mais objetos ativos (possivelmente definidos como uma composição). Em sistemas de tempo real, o comportamento concorrente dependente do estado complexo deve ser modelado usando uma cápsula contendo subcápsulas.

Se houver mais de 5 ± 2 estados em um único diagrama, considere o uso de subestados. O bom senso se aplica: dez estados em um padrão absolutamente regular podem ser suficientes, mas dois estados com quarenta transições entre eles obviamente precisam ser repensados. Certifique-se de que a máquina de estado seja compreensível.

Nomeie as transições para o que dispara o evento e/ou o que acontece durante a transição. Escolha nomes que melhorem a compreensão.

Quando você vê um vértice de escolha, você deve perguntar se pode delegar a responsabilidade por essa escolha a outro componente, de modo que seja apresentado ao objeto como um conjunto distinto de sinais a serem atuados (por exemplo, em vez de uma escolha no msg -> dados> x), faça com que o remetente ou algum outro ator intermediário tome a decisão e envie um sinal com a decisão explícita no nome do sinal (por exemplo, use sinais chamados Cheio e Vazio em vez de ter um sinal chamado valor e verificar os dados da mensagem).

Nomeie a pergunta respondida no vértice de escolha de forma descritiva, por exemplo, “AindaExisteVida ou “HoraDeReclamar”.

Dentro de qualquer objeto, tente manter os nomes dos vértices de escolha exclusivos (pela mesma razão de manter os nomes de transição exclusivos).

Existem fragmentos de código excessivamente longos nas transições? Em vez disso, as funções devem ser usadas e os fragmentos de código comuns são capturados como funções? Uma transição deve ser lida como um pseudocódigo de alto nível e deve seguir as mesmas ou até mais rígidas regras de comprimento que as funções C ++. Por exemplo, uma transição com mais de 25 linhas de código é considerada excessivamente longa.

As funções devem ser nomeadas de acordo com o que fazem.

Preste atenção especial às ações de entrada e saída: é particularmente fácil fazer alterações e esquecer de alterar as ações de entrada e saída.

As ações de saída podem ser usadas para fornecer recursos de segurança, por exemplo, a ação de saída do estado “Aquecedor” desliga o aquecedor, onde as ações são usadas para impor uma afirmação.

Geralmente, os subestados devem conter dois ou mais estados, a menos que a máquina de estados seja abstrata e seja refinada por subclasses do elemento envolvente.

Os pontos de escolha devem ser usados no lugar da lógica condicional em ações ou transições. Os pontos de escolha são facilmente vistos, ao passo que a lógica condicional no código fica oculta e fácil de ignorar.

Evite condições de guarda

- Se o evento disparar várias transições, não haverá controle sobre qual condição de guarda é avaliada primeiro. Como resultado, os resultados podem ser imprevisíveis.
- Mais de uma condição de guarda pode ser “verdadeira”, mas apenas uma transição pode ser seguida. O caminho escolhido pode ser imprevisível.
- As condições de proteção não são visuais; é mais difícil “ver” sua presença.

Evite máquinas de estado que se assemelham a fluxogramas

- Isso pode indicar uma tentativa de modelar uma abstração que não existe realmente, como:
  - Usar uma classe ativa para modelar o comportamento mais adequado para uma classe passiva (ou de dados) ou
  - Modelar uma classe de dados usando uma classe de dados e uma classe ativa que são fortemente acopladas (ou seja, a classe de dados foi usada para passar informações de tipo, mas a classe ativa contém a maioria dos dados que devem ser associados à classe de dados).
- Esse uso indevido de máquinas de estado pode ser reconhecido pelos seguintes sintomas
  - Mensagens enviadas para “eu”, principalmente apenas para reutilizar o código

    - Poucos estados, com muitos pontos de escolha
    - Em alguns casos, uma máquina de estado sem ciclos. Essas máquinas de estado são válidas em aplicativos de controle de processo ou ao tentar controlar uma sequência de eventos; sua presença durante a análise geralmente representa a degeneração da máquina de estado em um fluxograma.
- Quando o problema for identificado
    - Considere dividir a classe ativa em unidades menores com responsabilidades mais distintas,
    - Mova mais comportamento para uma classe de dados associada à classe ativa do problema.
    - Mova mais comportamento para funções de classe ativas.
    - Faça sinais mais significativos em vez de depender de dados.

7.3 – Componentes básicos de um diagrama de estados

Os componentes de um diagrama de estado são símbolos gráficos com funções específicas para representação apropriada do estado de um sistema. Eles foram desenvolvidos para ter simplicidade de uso. Esses símbolos podem ser vistos na figura 7.1.

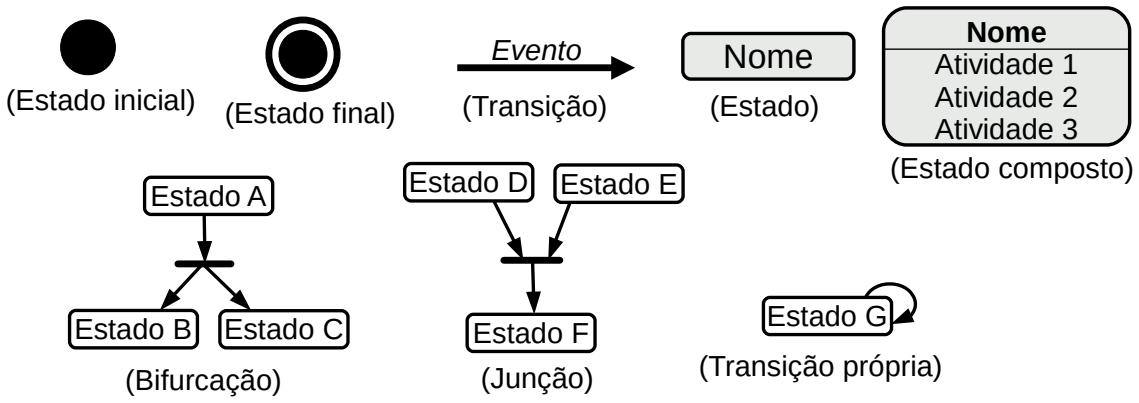

Figura 7.1 – Símbolos usados em diagramas de estados.

Estado inicial. Um círculo preto é usado para representar o estado inicial de um sistema ou classe.

Estado final. Um círculo preenchido dentro de um círculo é usado para representar o estado final em um diagrama de máquina de estado.

Transição. Uma seta sólida é usado para representar a transição ou mudança de controle de um estado para outro. A seta é rotulada com o evento que causa a mudança de estado. Esse evento também é chamado de "condição de guarda".

Estado. Um retângulo arredondado é usado para representar um estado. Um estado representa as condições ou circunstâncias de um objeto de uma classe em um instante de tempo.

Estado composto. Um retângulo arredondado é usado para representar também um estado composto. Representamos um estado com atividades internas usando um estado composto.

Bifurcação. Uma barra retangular sólida arredondada é usada para representar uma notação de bifurcação com a seta de entrada do estado pai e setas de saída em direção aos estados recém-criados. Usamos a notação de bifurcação para representar um estado dividido em dois ou mais estados simultâneos.

Junção. Uma barra retangular sólida arredondada é usada para representar uma notação de junção com setas de entrada dos estados de junção e seta de saída em direção ao estado de objetivo comum. Usamos

a notação de junção quando dois ou mais estados convergem simultaneamente em um na ocorrência de um evento ou eventos.

Transição própria. Uma seta sólida apontando para o estado em si para representar uma transição própria. Pode haver cenários em que o estado do objeto não muda com a ocorrência de um evento. Usamos auto-transições para representar tais casos.

7.4 – Desenhar um diagrama de estados

Passos para desenhar um diagrama de estado

- Identifique o estado inicial e os estados de terminação finais.
- Identifique os possíveis estados em que o objeto pode existir (valores de fronteira correspondentes a diferentes atributos nos orientam na identificação de diferentes estados).
- Identifique os eventos que acionam essas transições.

Exemplo. Diagrama de estado para um pedido online

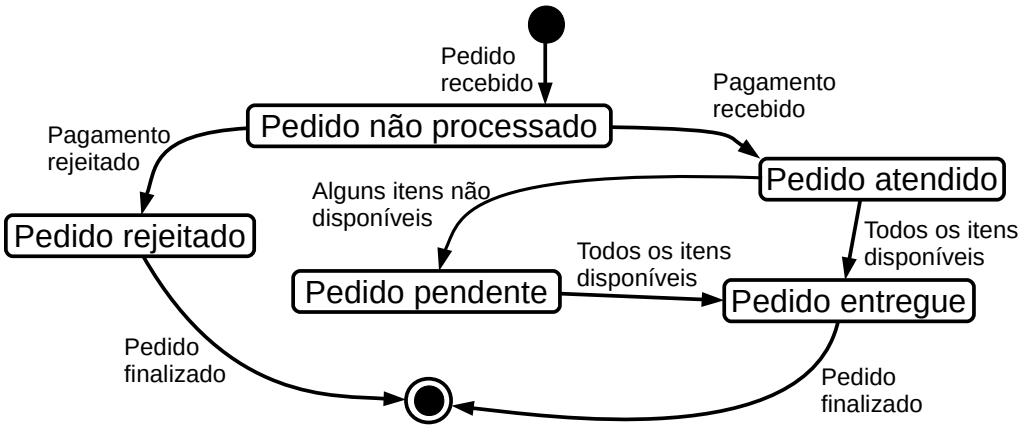

Figura 7.2 – Diagrama de estado para um pedido online.

Aqui está apenas um exemplo de como um sistema de pedidos online pode ser parecido com:

- No caso de um pedido ser recebido, nós transitamos de nosso estado inicial para o estado de “Pedido não processado”.
- O pedido não processado é então verificado.
- Se o pedido for rejeitado, passamos para o estado “Pedido rejeitado”.
- Se o pedido for aceito e tivermos os artigos disponíveis, transitamos para o estado de “Pedido entregue”.
- No entanto, se os itens não estiverem disponíveis, transitamos para o estado de “Pedido Pendente”.
- Depois que o pedido é entregue, nós transitamos para o estado final. Neste exemplo, mesclamos os dois estados, ou seja, “Pedido entregue” e “Pedido rejeitado” em um estado final.

Observação: Aqui também poderíamos ter tratado o “Pedido entregue” e o “Pedido rejeitado” como estados finais separadamente.

7.5 – Diagrama de estados em automação industrial

Para aplicar o método de diagrama de estados em máquinas ou processo com objetivo de automação, é necessário identificar os “estados” da máquina ou processo. Essa identificação é feita pela análise das condições da máquina ou processo comparado com as condições anteriores e posteriores. Num determinado intervalo de tempo, a máquina ou processo estão com os dispositivos de entrada e saída numa determinada condição, se digital, em ligado ou desligado.

7.5.1 – Exemplo 1. Portão automático

Automatizar um portão eletrônico do tipo basculante com sensor de contato para que, quando o portão estiver fechando e encontrar um obstáculo, o portão reverta a ação de fechar. Considere um temporizador de 60 segundos para manter o portão aberto, depois do tempo transcorrido o portão fecha. O portão não pode parar no meio do curso, ou seja, o portão somente fica parado em aberto ou em fechado. Existe apenas um botão para acionamento e o motor é acionado por 2 (dois) relés, um relé para abrir e outro relé para fechar. Quando o controlador for ligado o portão deve fechar imediatamente. Existem sensores que identificam que o portão está totalmente aberto e totalmente fechado.

Solução:

Para proceder com a solução deste problema, deve-se identificar os estados da máquina. Esses estados são as condições de funcionamento da máquina, onde o controlador está executando uma ação sobre os elementos de atuação. O controlador também pode estar apenas esperando um evento externo para evoluir para outro estado.

Os seguintes estados, ou condições da máquina, são identificados:

1. O portão está fechado e nenhuma ação está ocorrendo.
2. O portão está abrindo e o relé do motor para abrir está ativado.
3. O portão está aberto e um temporizador de 1 minuto está ativado.
4. O portão está fechando e o relé do motor para fechar está ativado.

Esses estados são então nomeados e dispostos em uma estrutura gráfica, que pode ser visto na figura 7.3. Observe que os nomes dos estados são representativos do comportamento da máquina. Alguns estados possuem ações associadas. O estado de nome “Fechado” não possui ações e portanto é um estado de espera.

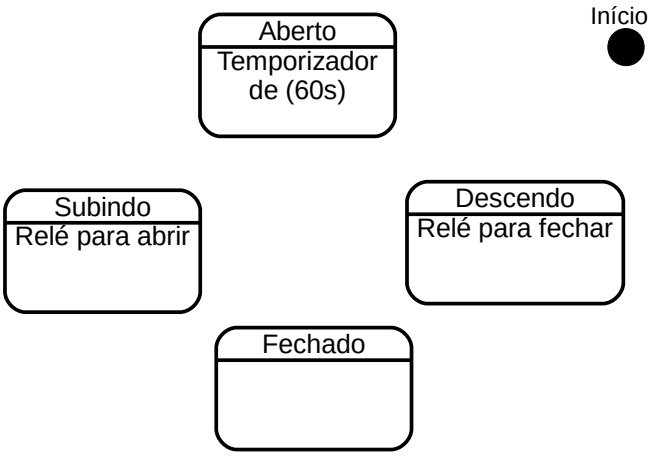

Figura 7.3 – Diagrama de estados para o portão automático (sem as transições).

O próximo passo é identificar as transições, ou condições de mudança, que permite a desativação de um estado e ativação de outro estado. As ações associadas a um estado são ativadas quando o estado é ativado, e estas ações são desativadas quando o estado é desativado.

As transições podem ser identificadas como

A condição de partida do controlador implica na transição do estado “Início” para o estado “Descendo”.

Com o portão no estado “Descendo”, o relé do motor para descer está ativado, o portão move-se para baixo pela ação do motor. Então pode acontecer duas possibilidades: o portão chegar totalmente embaixo ou o portão encontrar um obstáculo antes de chegar em baixo. Se o sensor de fim-de-curso com etiqueta “baixo” for ativado implica na transição do estado “Descendo” para o estado “Fechado”. Se o sensor de fim-de-curso com etiqueta “obstáculo” for ativado implica na transição do estado “Descendo” para o estado “Subindo”.

Com o portão fechado, o controlador fica esperando o usuário apertar o “Botão” e isso implica na transição do estado “Fechado” para o estado “Subindo”.

Com o portão no estado “Subindo”, o relé do motor para subir está ativado, o portão move-se para cima pela ação do motor. Quando o sensor de fim-de-curso com etiqueta “alto” for ativado implica na transição do estado “Subindo” para o estado “Aberto”.

Com o portão aberto, o controlador ativa um temporizador de 1 minuto (ou 60 segundos) e fica aguardando ou o término da contagem de tempo do temporizador ou o usuário apertar o “Botão” e isso implica na transição do estado “Aberto” para o estado “Fechando”.

Essas transições são colocadas no diagrama de estados na forma de linhas orientadas em seta com etiquetas que são as condições de mudança de estado da máquina. O formato das linhas orientadas não é formalizado, sendo aceitável linhas retas, linhas com ângulos retos, linhas de curvas simples e linhas de curvas múltiplas.

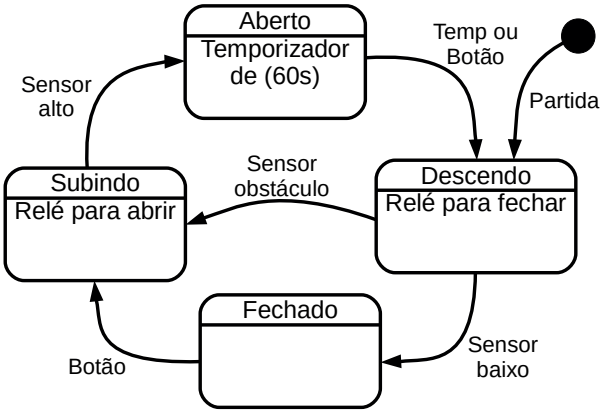

Figura 7.4 – Diagrama de estados para o portão automático.

Para fazer a codificação em linguagem Ladder procede-se de forma semelhante aos outros métodos, com as 3 regiões: inicialização, lógica e saídas. A seção de inicialização pode ser automática ou através de um comando local ou remoto para a partida da máquina. A seção de lógica é realizada pela ativação e desativação dos estados de acordo com a lógica da transição. As saídas são ativadas com a ativação dos estados correspondentes.

A tabela de entradas é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 7.1.

Parafuso	Etiqueta	Descrição
I01	PTD	Botoeira NA. (“Partida”)
I02	BTN	Botoeira NA. (“Botão”)
I03	ALT	Sensor fim-de-curso eletromecânico. (“Alto”)
I04	BXO	Sensor fim-de-curso eletromecânico. (“Baixo”)
I05	OBT	Sensor fim-de-curso eletromecânico. (“Obstáculo”)

Tabela 7.1 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 7.2.

Parafuso	Etiqueta	Descrição
Q01	SBR	Relé de acionamento de motor de indução. (“Subir”)
Q02	DCR	Relé de acionamento de motor de indução. (“Descer”)

Tabela 7.2 – Descrição das saídas.

Com essas etiquetas reduzidas, é necessário refazer o diagrama de estados. Esse diagrama de estados pode ser visto na figura 7.5. Observe que a “Partida” foi substituído pelo contato especial de primeiro ciclo (“first scan” - “F.S.”).

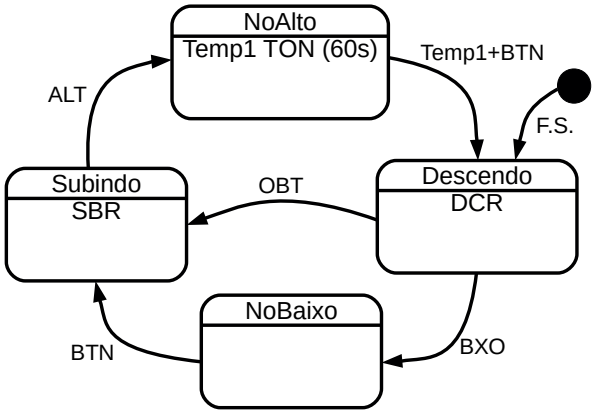

Figura 7.5 – Diagrama de estados para o portão automático (com etiquetas reduzidas).

Codificação do diagrama de estados na linguagem Ladder

A codificação em linguagem Ladder para este diagrama de estados pode ser visto nas figuras 7.6 a 7.11.

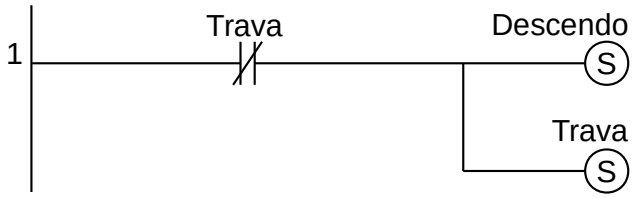

Figura 7.6 – Programa em linguagem Ladder para a transição de inicialização.

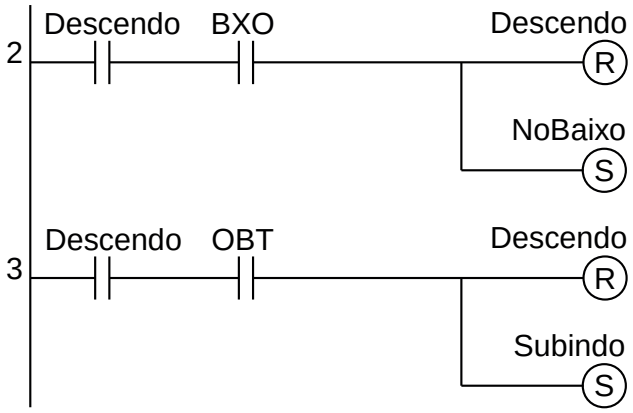

Figura 7.7 – Programa em linguagem Ladder para as transições saintes do estado Descendo.

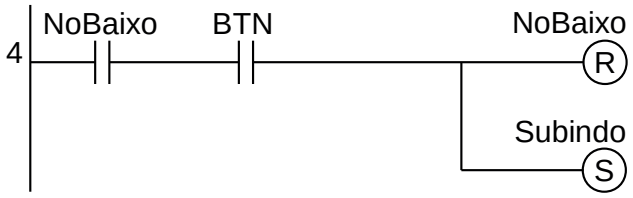

Figura 7.8 – Programa em linguagem Ladder para as transições saintes do estado NoBaixo.

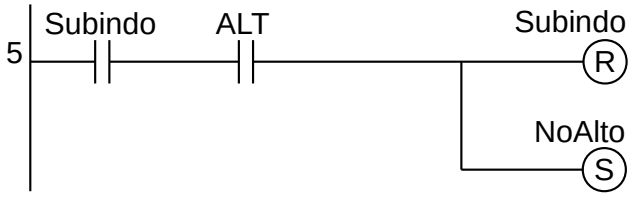

Figura 7.9 – Programa em linguagem Ladder para as transições saintes do estado Subindo.

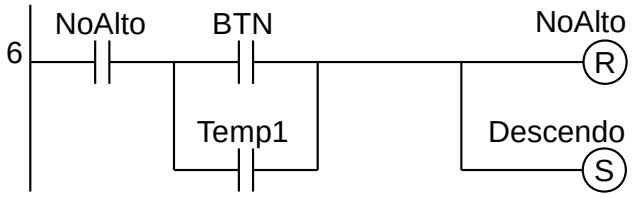

Figura 7.10 – Programa em linguagem Ladder para as transições saintes do estado NoAlto.

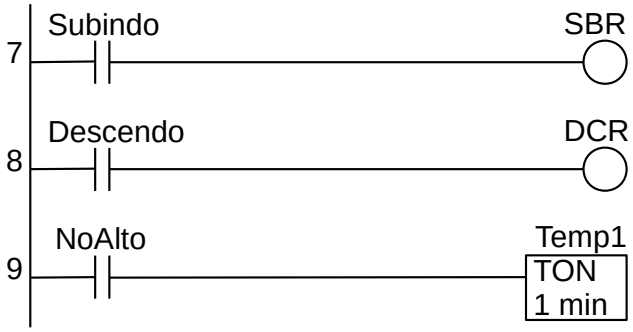

Figura 7.11 – Programa em linguagem Ladder para as saídas.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 7.1 e 7.2. A tabela com as variáveis internas pode ser vista na tabela 7.3.

Variável	Etiqueta	Descrição
M01	Subindo	Memória para estado.
M02	Descendo	Memória para estado.
M03	NoAlto	Memória para estado.
M04	NoBaixo	Memória para estado.
M05	Trava	Memória para trava de inicialização.
T01	Temp1	Temporizador para ligar (TON) de 60 segundos.

Tabela 7.3 – Descrição das variáveis internas.

7.5.2 – Exemplo 2. Controle de acesso de carros e motos

Esse exemplo mostra uma máquina mais complexa com várias decisões, que são estados que apresentam mais de uma saída.

Descrição da máquina

O sistema é composto de duas barreiras etiquetadas como “E” e “D” como se vê na figura 7.12. À esquerda das barreiras existe uma caixa de pagamento que pode receber moedas de 25 e 50 centavos. A moeda de 25 centavos ativa um sensor do tipo fim-de-curso com etiqueta “u”. A moeda de 50 centavos ativa um sensor do tipo fim-de-curso com etiqueta “v”. No solo existem duas placas com sensor de pressão com etiquetas “A” e “B” destinadas a detectar a presença de veículos. Considere que as barreias possuem sensores em cima (“Ea” e “Da”) e embaixo (“Eb” e “Db”) para identificar quando as barreiras estão abertas ou fechadas. Cada barreira é acionada por um motor de indução reversível através de dois relés para giro do motor para abrir (“MEa” e “MDa”) para fechar (“MEf” e “MDf”) a barreira. Existe um alarme sonoro na caixa de pagamento com etiqueta “Alm”. Considere um painel de comando com uma chave para ligar e desligar o sistema do tipo rotativo, uma botoeira para iniciar o funcionamento do sistema com etiqueta “PTD” para partida e outra botoeira para parar o funcionamento do sistema com etiqueta “PRD”. Não existem lâmpadas sinalizadoras.

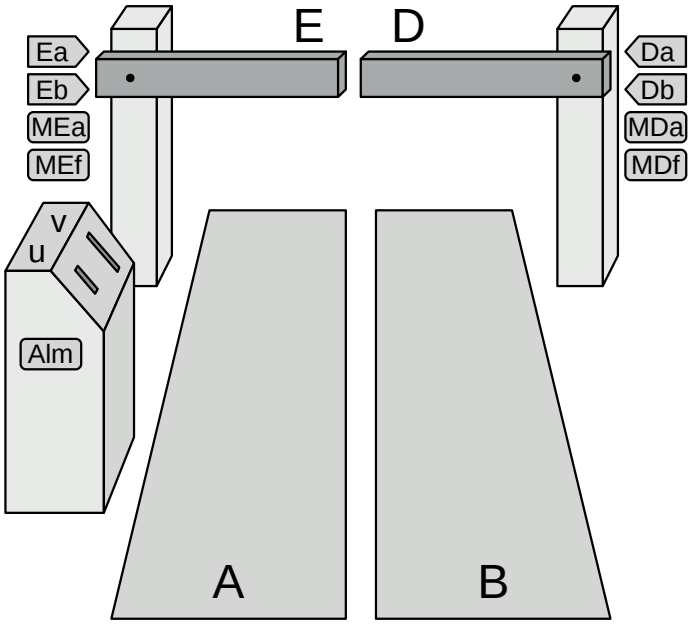

Figura 7.12 – Controle de acesso de carros e motos.

Descrição do funcionamento

A parte esquerda pode abrir-se isoladamente e deixar entrar veículos de duas rodas. Ambas as partes podem abrir-se em conjunto e deixar passar veículos de quatro rodas.

Para que se abra apenas a parte esquerda da barreira para passar veículos de 2 rodas, é necessário que um veículo se coloque inteiramente sobre "A" e coloque pelo menos uma moeda de 25 centavos em "u". Se uma moeda de 50 centavos for depositada em "v" o sistema deve aceitar o pagamento e troco não é devolvido. Quando um veículo de 2 rodas se posiciona em "B" o alarme sonoro deve tocar e somente parar quando o veículo recuar e sair de sobre a placa "B". A barreira fecha-se quando o veículo de duas rodas abandona o sensor "A".

Quando um veículo de 4 rodas pretende entrar ele estará sobre as placas "A" e "B" e colocar pelo menos uma moeda de 50 centavos em "v" ou duas moedas de 25 centavos em "u". Se uma moeda de 25 centavos for depositada em "u" e depois uma moeda de 50 centavos for depositada em "v" o sistema deve aceitar o pagamento e troco não é devolvido. A barreira fecha-se quando o veículo abandona os sensores "A" e "B". Notar que o caracteriza um veículo de quatro rodas é o fato de "A" e "B" serem tocados dentro de um intervalo de tempo inferior a 2s.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entradas é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 7.4.

Parafuso	Etiqueta	Descrição
I01	PTD	Botoeira NA. (Partida)
I02	PRD	Botoeira NA. (Parada)
I03	A	Sensor de pressão, fim-de-curso e mola.
I04	B	Sensor de pressão, fim-de-curso e mola.
I05	u	Sensor fim-de-curso. (Moeda de 25 centavos)
I06	v	Sensor fim-de-curso. (Moeda de 50 centavos)
I07	Ea	Sensor fim-de-curso. (Barreira esquerda no alto)
I08	Eb	Sensor fim-de-curso. (Barreira esquerda no baixo)
I09	Da	Sensor fim-de-curso. (Barreira direita no alto)
I10	Db	Sensor fim-de-curso. (Barreira direita no baixo)

Tabela 7.4 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 7.5.

Parafuso	Etiqueta	Descrição
Q01	MEa	Contator para motor de indução. (Barreira esquerda abrir)
Q02	MEf	Contator para motor de indução. (Barreira esquerda fechar)
Q03	MDa	Contator para motor de indução. (Barreira direita abrir)
Q04	MDf	Contator para motor de indução. (Barreira direita fechar)
Q05	Alm	Sinalizador sonoro. (Alarme)

Tabela 7.5 – Descrição das saídas.

Condição inicial

Considere que as barreiras estão abaixadas, ou seja, os sensores de barreira aberta (“Ea” e “Da”) em falso e os sensores de barreira fechada (“Eb” e “Db”) em verdadeiro. Não tem carro ou moto na entrada do acesso, ou seja, “A” e “B” em falso. Nenhuma moeda foi inserida, ou seja, “u” e “v” em falso.

Parte 1 – Inicialização. Solução por frases lógicas

É criado a variável “Func” que indica que o programa do controlador pode ser executado.

Esta variável é controlada pelas botoeiras de partida “PTD” e de parada “PRD”. Também é criada a variável “Trava” para inibir o reinício do primeiro bloco do diagrama de estados através da botoeira de inicialização.

É utilizado frases lógicas para a modelagem dessa etapa.

A primeira frase produz um “pulso único” da botoeira “PTD”. Esse pulso ativa o primeiro bloco do diagrama de estados.

SE (“PTD” é verdadeiro e “Trava” é falso) ENTÃO (fazer “Espera1” verdadeiro e memorizar e fazer “Trava” verdadeiro e memorizar)

A segunda frase permite o reinício do programa, habilitando o funcionamento do programa.

SE (“PTD” é verdadeiro) ENTÃO (fazer a variável “Func” verdadeiro e memorizar).

A terceira frase desabilita a variável de funcionamento do programa.

SE (“PRD” é verdadeiro) ENTÃO (fazer a variável “Func” falso).

Parte 2 – Funcionamento. Solução por diagrama de estados

O diagrama de estados para solucionar este problema pode ser visto na figura 7.13.

Detalhamento dos estados e transições

Estado inicial: energização.

O controlador é energizado e fica apto a executar o programa.

Transição inicial: inicio → “Espera1” Lógica: “PTD”.

O estado “Espera1” é ativado.

Identificação do veículo

Estado: “Espera1”

Nenhuma ação é realizada, o controlador fica aguardando a chegada de um veículo.

Transição: “Espera1” → “Tempo1” Lógica: “(A+B).Func”

Um veículo ativa um dos sensores de presença de veículos “A” ou “B”. A variável “Func” deve estar ativa (verdadeira). O estado “Espera1” é desativado e o estado “Tempo1” é ativado.

Estado: “Tempo1”

Um temporizador “Temp1” para ligar (TON) de 2 segundos é ativado.

Transição: “Tempo1” → “Carro” Lógica: “(A.B.temp1)”

O temporizador terminou a contagem estabelecida. Um veículo de 4 rodas está sobre os sensores “A” e “B”. O estado “Tempo1” é desativado e o estado “Carro” é ativado.

Transição: “Tempo1” → “Moto” Lógica: “($A.\overline{B}$.temp1)”

O temporizador terminou a contagem estabelecida. Um veículo de 2 rodas está sobre o sensor “A”. O estado “Tempo1” é desativado e o estado “Moto” é ativado.

Transição: “Tempo1” → “Alarme” Lógica: “($\overline{A}$.B.temp1)”

O temporizador terminou a contagem estabelecida. Um veículo de 2 rodas está sobre o sensor “B”. O estado “Tempo1” é desativado e o estado “Alarme” é ativado.

Recepção de moedas para o carro

Estado: “Carro”

Nenhuma ação é realizada, o controlador fica aguardando a colocação de moedas.

Transição: “Carro” → “AbreCarro” Lógica: “v”

O motorista do veículo de 4 rodas depositou uma moeda de 50 centavos.

O estado “Carro” é desativado e o estado “AbreCarro” é ativado.

Transição: “Carro” → “MaisMoeda” Lógica: “u”

O motorista do veículo de 4 rodas depositou uma moeda de 25 centavos.

O estado “Carro” é desativado e o estado “MaisMoeda” é ativado.

Estado: “MaisMoeda”

Nenhuma ação é realizada, o controlador fica aguardando a colocação de moedas.

Transição: “MaisMoeda” → “AbreCarro” Lógica: “u+v”

O motorista do veículo de 4 rodas depositou uma moeda de 25 centavos ou uma moeda de 50 centavos. O estado “MaisMoeda” é desativado e o estado “AbreCarro” é ativado.

Recepção de moedas para a moto

Estado: “Moto”

Nenhuma ação é realizada, o controlador fica aguardando a colocação de moedas.

Transição: “Moto” → “AbreMoto” Lógica: “u+v”

O motorista do veículo de 2 rodas depositou uma moeda de 25 centavos ou uma moeda de 50 centavos. O estado “Moto” é desativado e o estado “AbreMoto” é ativado.

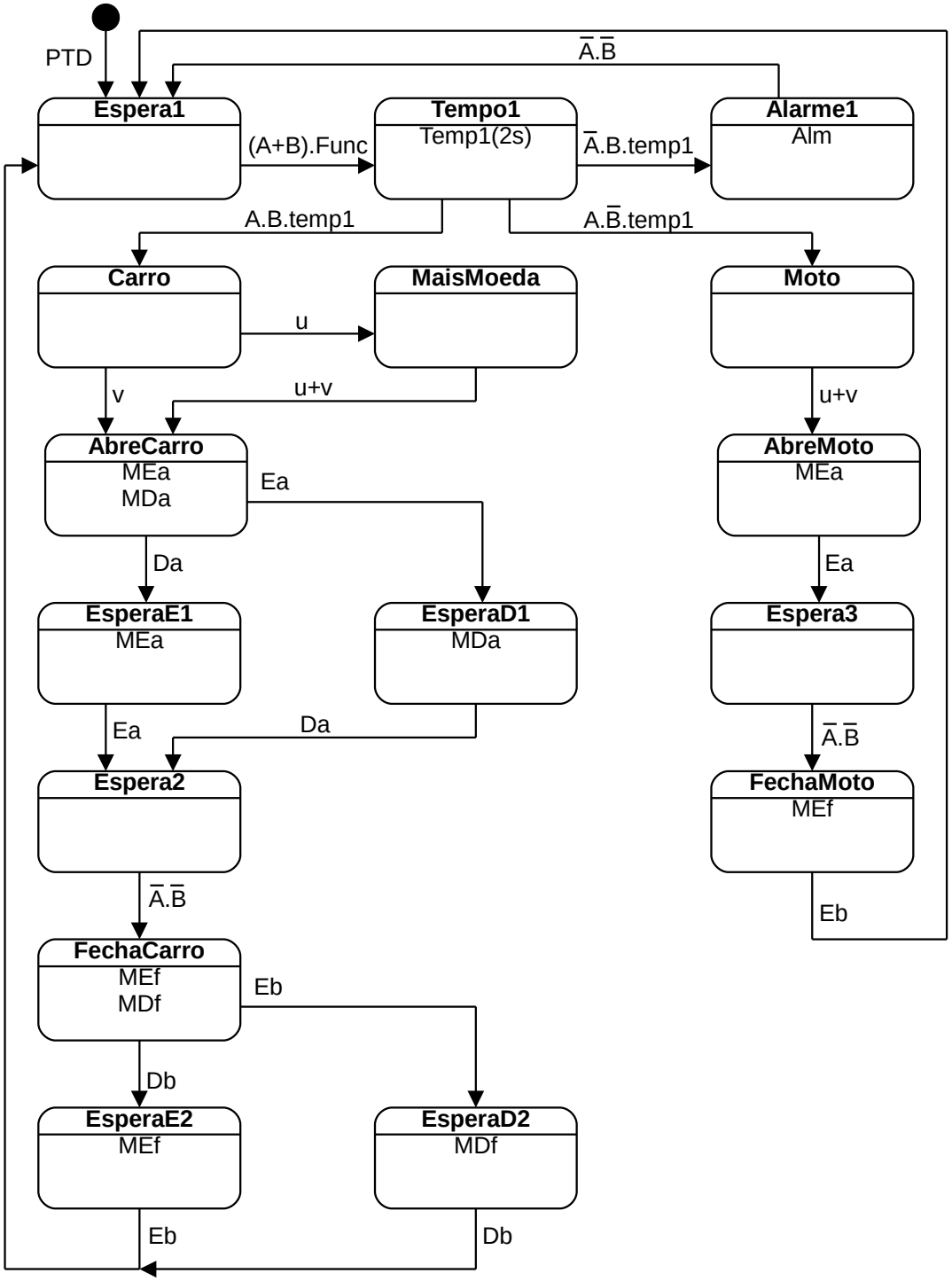

Figura 7.13 – Diagrama de estados para o controle de acesso de carros e motos.

Ativação do alarme

Estado: “Alarme”

A saída “Alm” é ativada.

Transição: “Alarme” → “Espera1” Lógica: $\overline{A}.\overline{B}$

O veículo de 2 rodas sai de sobre o sensor “B”.

O estado “Alarme” é desativado e o estado “Espera1” é ativado.

Abertura e fechamento das barreiras para passagem do carro

Estado: “AbreCarro”

Os motores das barreiras “E” e “D” são ativados para abrir, ou seja, as saídas “MEa” e “MDa” são ativadas.

Transição: “AbreCarro” → “EsperaD1” Lógica: “Ea”

A barreira “E” chegou no alto, totalmente aberta, primeiro que a barreira “D”.

O estado “AbreCarro” é desativado e o estado “EsperaD1” é ativado.

Estado: “EsperaD1”

A saída “MDa” continua ativada.

Transição: “EsperaD1” → “Espera2” Lógica: “Da”

A barreira “D” chegou no alto, totalmente aberta.

O estado “EsperaD1” é desativado e o estado “Espera2” é ativado.

Transição: “AbreCarro” → “EsperaE1” Lógica: “Da”

A barreira “D” chegou no alto, totalmente aberta, primeiro que a barreira “E”.

O estado “AbreCarro” é desativado e o estado “EsperaE1” é ativado.

Estado: “EsperaE1”

A saída “MEa” continua ativada.

Transição: “EsperaE1” → “Espera2” Lógica: “Ea”

A barreira “E” chegou no alto, totalmente aberta.

O estado “EsperaE1” é desativado e o estado “Espera2” é ativado.

Estado: “Espera2”

Nenhuma ação é realizada, o controlador fica aguardando a saída do veículo de 4 rodas.

Transição: “Espera2” → “FechaCarro” Lógica: $\overline{A}.\overline{B}$

O veículo de 4 rodas sai de sobre os sensores “A” e “B”.

O estado “Espera2” é desativado e o estado “FechaCarro” é ativado.

Estado: “FechaCarro”

Os motores das barreiras “E” e “D” são ativados para fechar, ou seja, as saídas “MEf” e “MDf” são ativadas.

Transição: “FechaCarro” → “EsperaD2” Lógica: “Eb”

A barreira “E” chegou em baixo, totalmente fechada, primeiro que a barreira “D”.

O estado “FechaCarro” é desativado e o estado “EsperaD2” é ativado.

Estado: “EsperaD2”

A saída “MDf” continua ativada.

Transição: “EsperaD2” → “Espera1” Lógica: “Db”

A barreira “D” chegou em baixo, totalmente fechada.

O estado “EsperaD2” é desativado e o estado “Espera1” é ativado.

Transição: “FechaCarro” → “EsperaE2” Lógica: “Db”

A barreira “D” chegou em baixo, totalmente fechada, primeiro que a barreira “E”.

O estado “FechaCarro” é desativado e o estado “EsperaE2” é ativado.

Estado: “EsperaE2”

A saída “MEf” continua ativada.

Transição: “EsperaE2” → “Espera1” Lógica: “Eb”

A barreira “E” chegou em baixo, totalmente fechada.

O estado “EsperaE2” é desativado e o estado “Espera1” é ativado.

Abertura e fechamento da barreira para passagem da moto

Estado: “AbreMoto”

O motor da barreira “E” é ativado para abrir, ou seja, a saída “MEa” é ativada.

Transição: “AbreMoto” → “Espera3” Lógica: “Ea”

A barreira “E” chegou no alto, totalmente aberta.

O estado “AbreMoto” é desativado e o estado “Espera3” é ativado.

Estado: “Espera3”

Nenhuma ação é realizada, o controlador fica aguardando a saída do veículo de 2 rodas.

Transição: “Espera3” → “FechaMoto” Lógica: $\overline{A}.\overline{B}$

O veículo de 2 rodas sai de sobre os sensores “A” e “B”.

O estado “Espera3” é desativado e o estado “FechaMoto” é ativado.

Estado: “FechaMoto”

O motor da barreira “E” é ativado para fechar, ou seja, a saída “MEf” é ativada.

Transição: “FechaMoto” → “Espera1” Lógica: “Eb”

A barreira “E” chegou em baixo, totalmente fechada.

O estado “FechaMoto” é desativado e o estado “Espera1” é ativado.

Codificação do diagrama de estados em linguagem Ladder

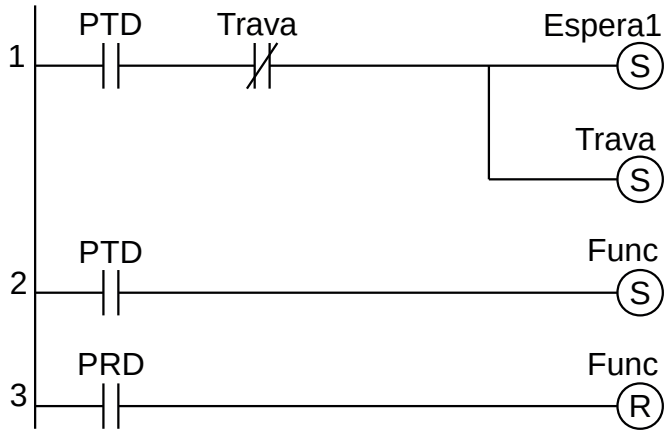

Figura 7.14 – Programa em linguagem Ladder para a inicialização.

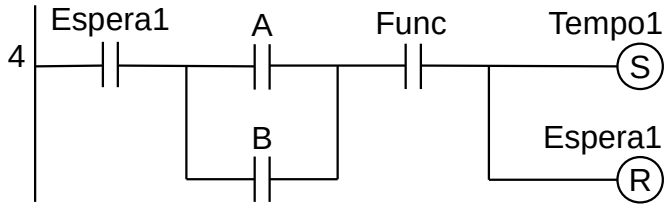

Figura 7.15 – Programa em linguagem Ladder para as transições saintes do estado Espera1.

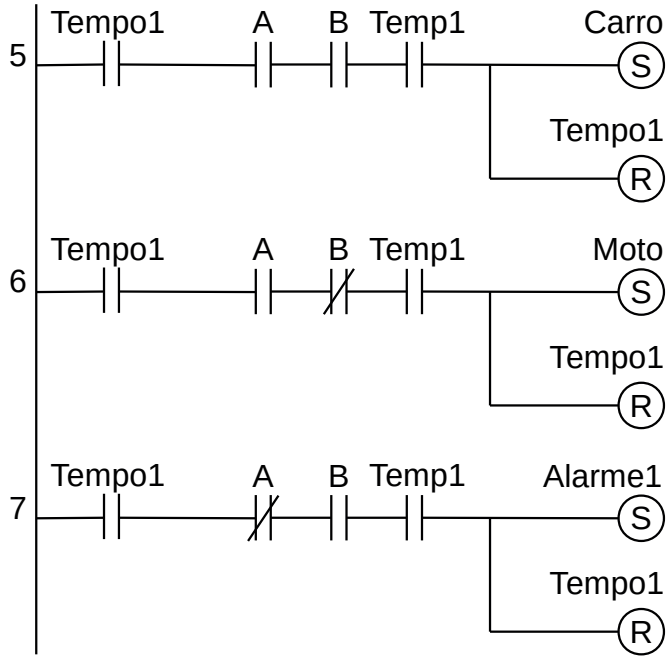

Figura 7.16 – Programa em linguagem Ladder para as transições saintes do estado Tempo1.

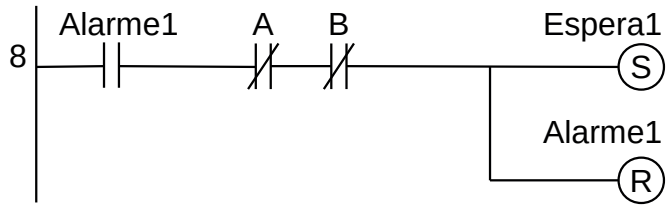

Figura 7.17 – Programa em linguagem Ladder para as transições saintes do estado Alarme1.

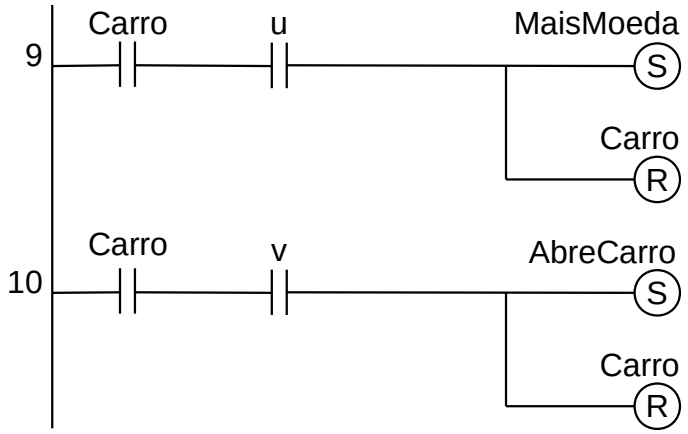

Figura 7.18 – Programa em linguagem Ladder para as transições saintes do estado Carro.

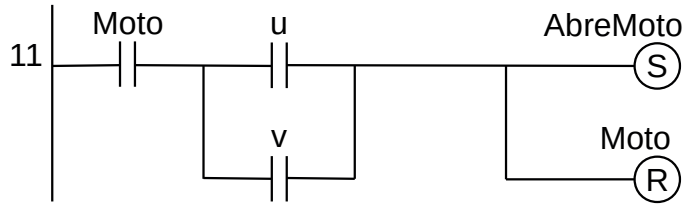

Figura 7.19 – Programa em linguagem Ladder para as transições saintes do estado Moto.

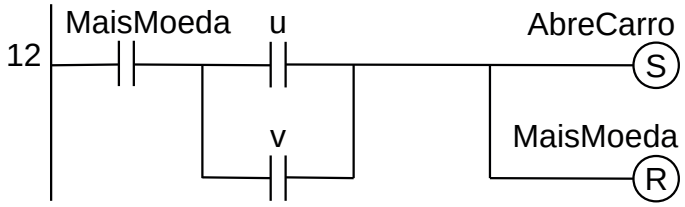

Figura 7.20 – Programa em linguagem Ladder para as transições saintes do estado MaisMoeda.

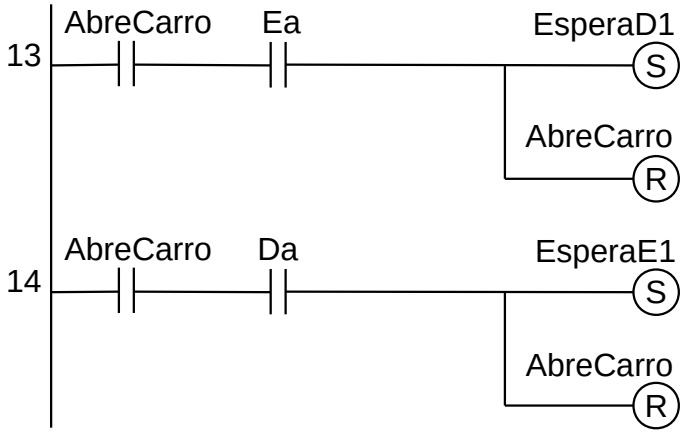

Figura 7.21 – Programa em linguagem Ladder para as transições saintes do estado AbreCarro.

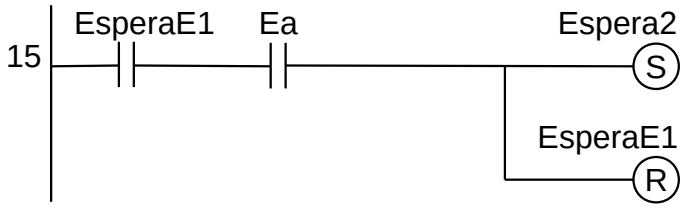

Figura 7.22 – Programa em linguagem Ladder para as transições saintes do estado EsperaE1.

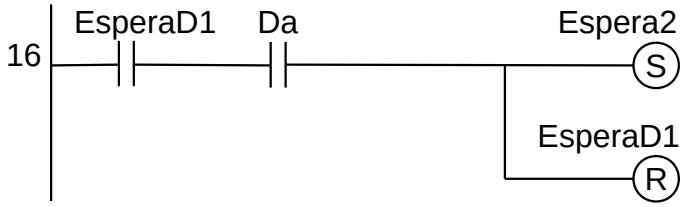

Figura 7.23 – Programa em linguagem Ladder para as transições saintes do estado EsperaD1.

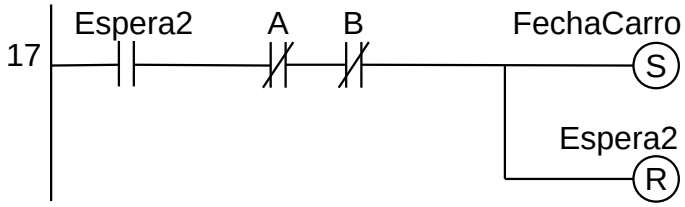

Figura 7.24 – Programa em linguagem Ladder para as transições saintes do estado Espera2.

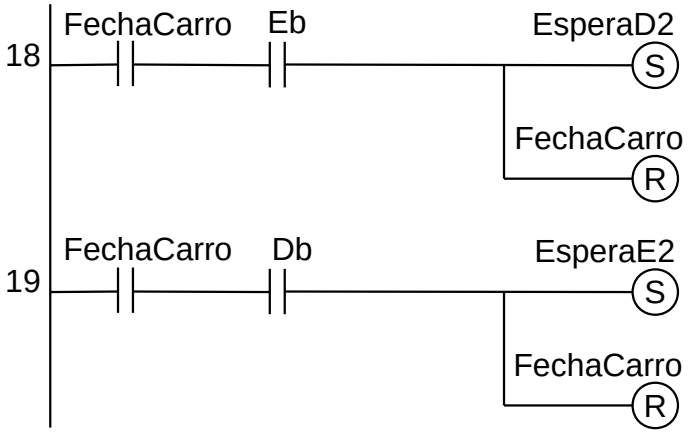

Figura 7.25 – Programa em linguagem Ladder para as transições saintes do estado FechaCarro.

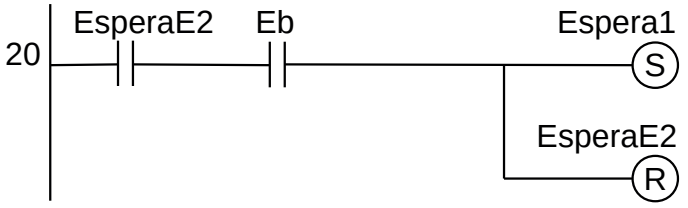

Figura 7.26 – Programa em linguagem Ladder para as transições saintes do estado EsperaE2.

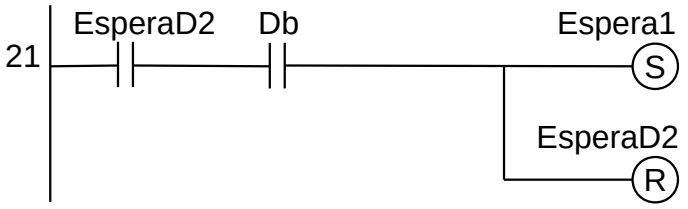

Figura 7.27 – Programa em linguagem Ladder para as transições saintes do estado EsperaD2.

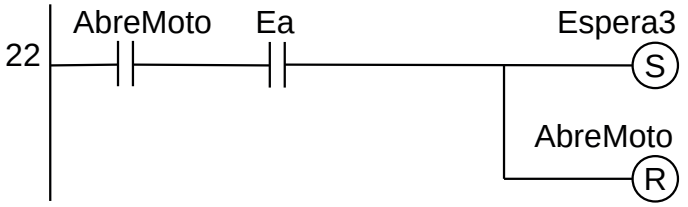

Figura 7.28 – Programa em linguagem Ladder para as transições saintes do estado AbreMoto.

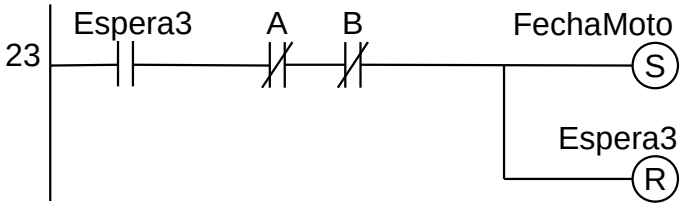

Figura 7.29 – Programa em linguagem Ladder para as transições saintes do estado Espera3.

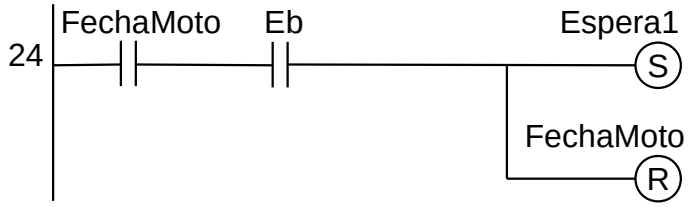

Figura 7.30 – Programa em linguagem Ladder para as transições saintes do estado FechaMoto.

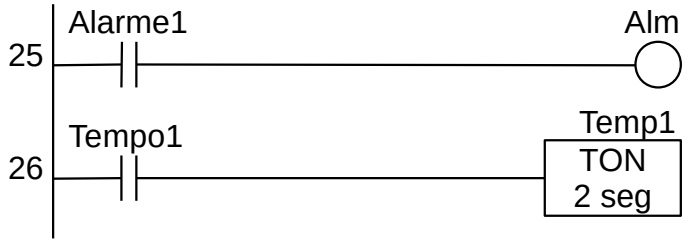

Figura 7.31 – Programa em linguagem Ladder para ativação das saídas de Alm e Temp1.

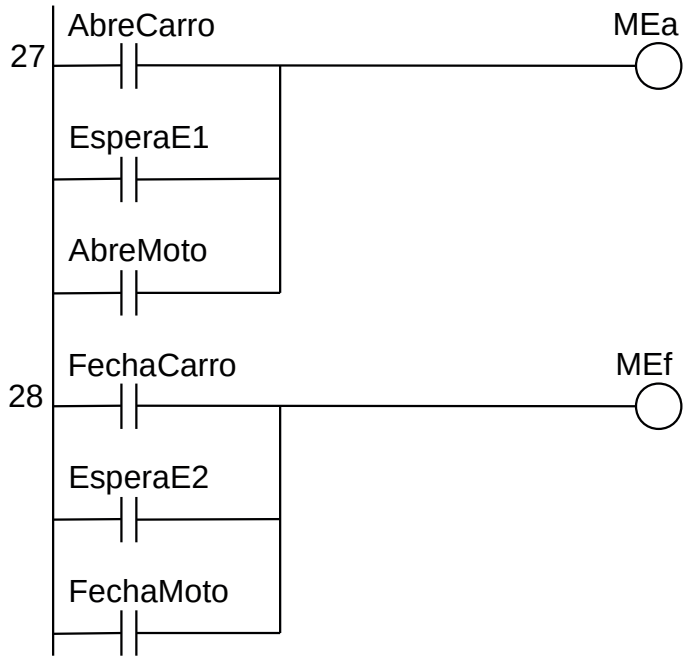

Figura 7.32 – Programa em linguagem Ladder para ativação das saídas de MEa e MEf.

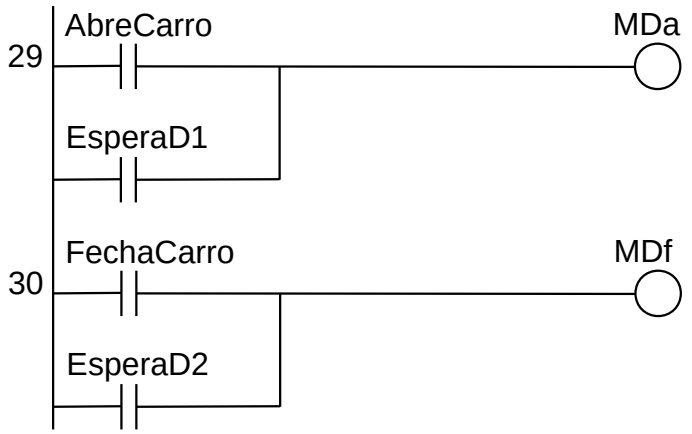

Figura 7.33 – Programa em linguagem Ladder para ativação das saídas de MDa e MDf.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 7.4 e 7.5. A tabela com as variáveis internas pode ser vista na tabela 7.6 .

Variável	Etiqueta	Descrição
M01	Espera1	Memória para estado.
M02	Tempo1	Memória para estado.
M03	Alarme1	Memória para estado.
M04	Carro	Memória para estado.
M05	Moto	Memória para estado.
M06	MaisMoeda	Memória para estado.
M07	AbreCarro	Memória para estado.
M08	EsperaD1	Memória para estado.
M09	EsperaE1	Memória para estado.
M10	Espera2	Memória para estado.
M11	FechaCarro	Memória para estado.
M12	EsperaD2	Memória para estado.
M13	EsperaE2	Memória para estado.
M14	AbreMoto	Memória para estado.
M15	Espera3	Memória para estado.
M16	FechaMoto	Memória para estado.
M17	Func	Memória de funcionamento do programa.
M18	Trava	Memória de trava de inicialização.
T01	Temp1	Temporizador para ligar (TON) 2 segundos.

Tabela 7.6 – Descrição das variáveis internas.

7.5.3 – Exemplo 3. Máquina de embalagem de produtos

Esse exemplo mostra a solução de um problema por meio de vários diagramas de estado interligados por variáveis e com vários estados de espera.

Descrição da máquina

Uma estação de embalagem de produtos, mostrada nas figuras 7.34, 7.35 e 7.36, deve ser automatizada. A estação possui os seguintes elementos:

- Esteira transportadora por onde chegam os produtos individualmente.
- Cilindro B de empilhamento, onde os produtos vão se agrupando.
- Cilindro A que introduz os produtos na caixa.
- Plataforma giratória acionada pelo cilindro C que levanta e coloca a caixa nos roletes de saída. A caixa é colocada manualmente e o operador aperta um botão “p”.
- Trava de empilhamento que garante que os fardos sejam formados.
- Existe uma botoeira de início do programa com etiqueta “Partida”.

Descrição do funcionamento

O sistema funciona do seguinte modo:

1. O operador da máquina coloca manualmente uma caixa no suporte do cilindro C e pressiona o botão “p”.
2. Os produtos chegam individualmente pela esteira e vão se acumulando sobre a plataforma de levantamento acionada pelo cilindro B.
3. Quando o primeiro produto que chegar ativar o sensor “t1” é o sinal que indica que a fileira de 3 produtos está completa. O cilindro B levanta a fileira de produtos por dentro da “trava de empilhamento”. A posição do cilindro B é determinada pelos sensores de fim-de-curso “b0” e “b1”. Esta operação se repete, 3 vezes, até a altura do fardo atingir o sensor “t2” que indica um fardo de 9 produtos está pronto.
4. Quando o fardo estiver pronto o cilindro A empurra os produtos para dentro da caixa. O cilindro B serve de guia. O cilindro A possui os sensores “a0”, “a1” e “a2”.
5. Quando a caixa estiver cheia o cilindro C a coloca sobre os roletes de saída.

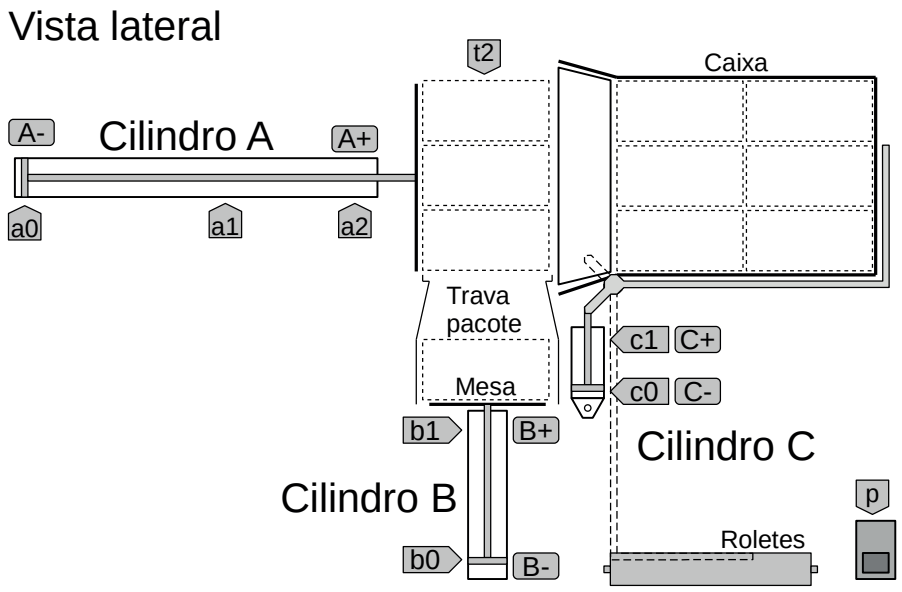

Figura 7.34 – Máquina de empacotamento. Vista lateral.

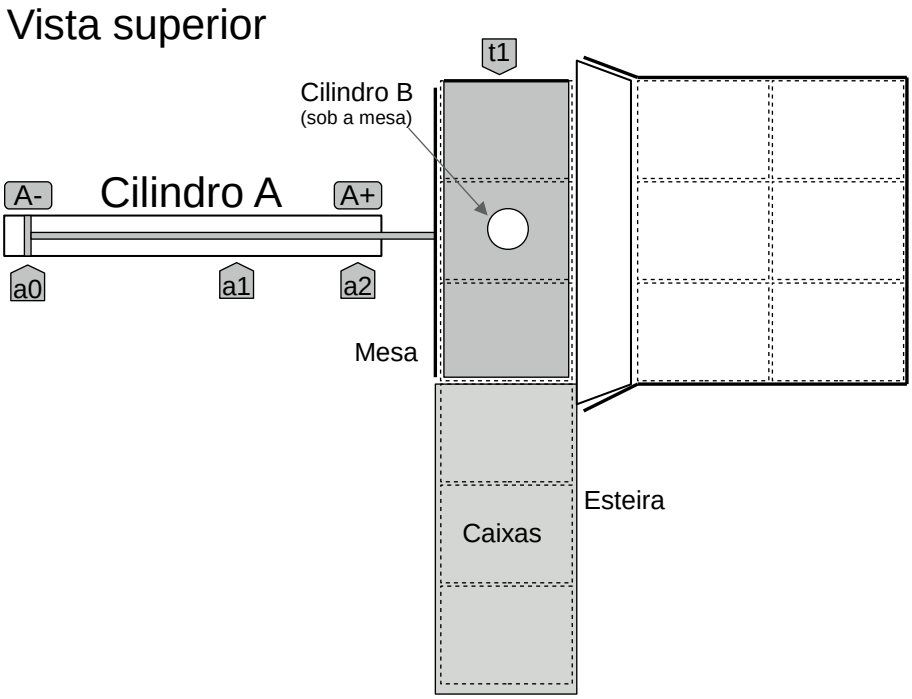

Figura 7.35 – Máquina de empacotamento. Vista superior.

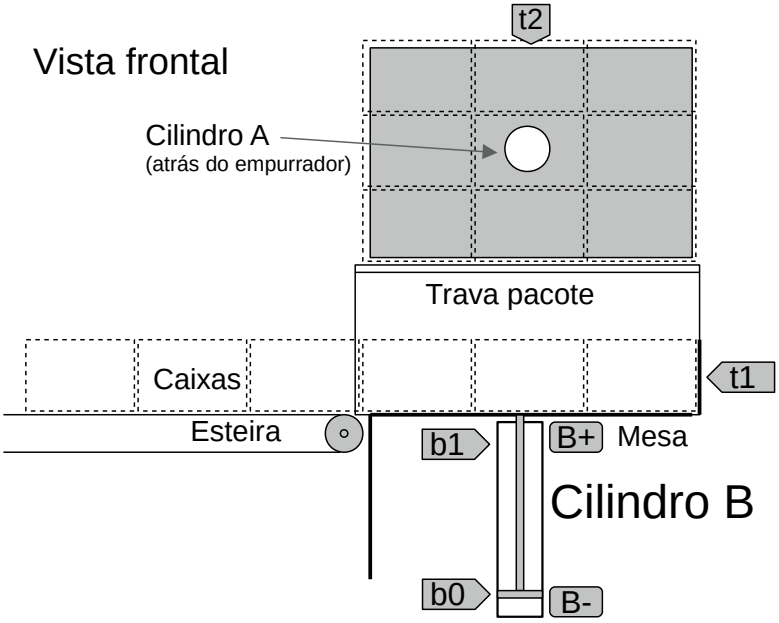

Figura 7.36 – Máquina de empacotamento. Vista frontal.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entradas é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 7.8.

Parafuso	Etiqueta	Descrição
I01	Partida	Botoeira NA.
I02	p	Botoeira NA.
I03	a0	Sensor tipo chave magnética.
I04	a1	Sensor tipo chave magnética.
I05	a2	Sensor tipo chave magnética.
I06	b0	Sensor tipo chave magnética.
I07	b1	Sensor tipo chave magnética.
I08	c0	Sensor tipo chave magnética.
I09	c1	Sensor tipo chave magnética.
I10	t1	Sensor fim-de-curso eletromecânico.
I11	t2	Sensor fim-de-curso eletromecânico.

Tabela 7.7 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 7.9.

Parafuso	Etiqueta	Descrição
Q01	A+	Eletroválvula para cilindro pneumático.
Q02	A-	Eletroválvula para cilindro pneumático.
Q03	B+	Eletroválvula para cilindro pneumático.
Q04	B-	Eletroválvula para cilindro pneumático.
Q05	C+	Eletroválvula para cilindro pneumático.
Q06	C-	Eletroválvula para cilindro pneumático.

Tabela 7.8 – Descrição das saídas.

Solução por diagrama de estados

O diagrama de estados para este problema pode ser visto na figura 7.37.

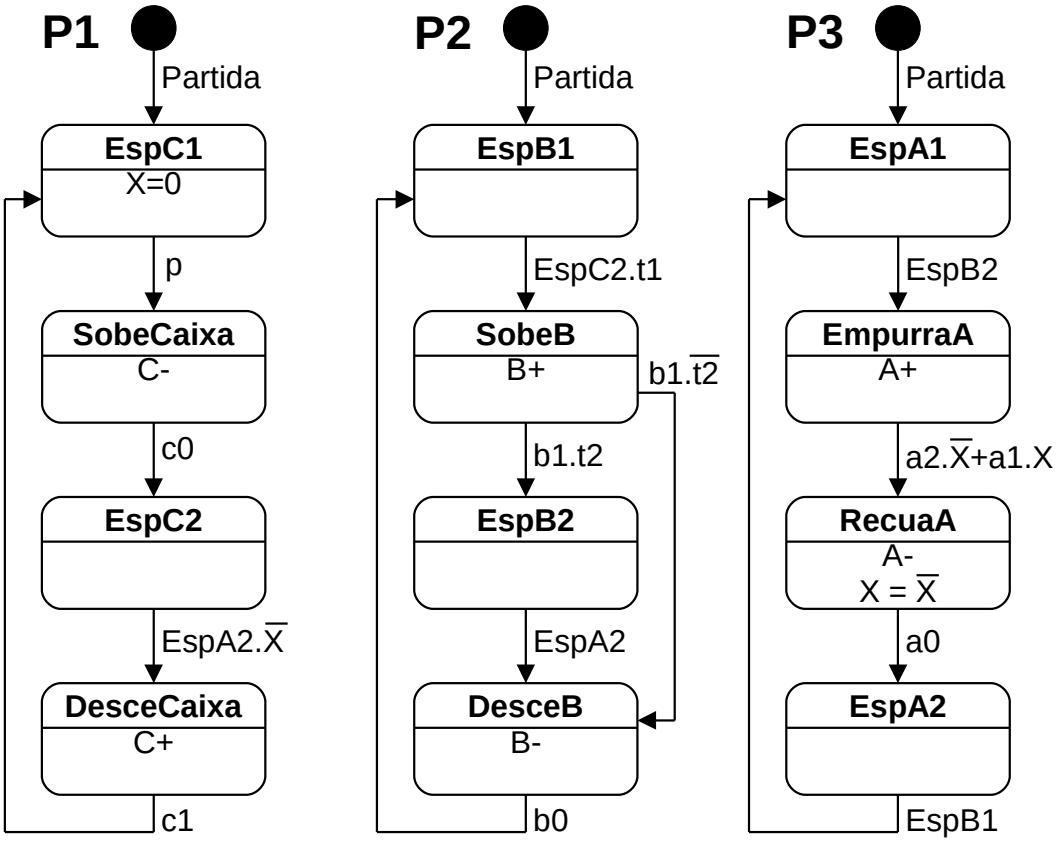

Figura 7.37 – Diagrama de estados para a máquina de empacotamento.

Detalhamento dos estados e transições

Estado inicial: energização.

O controlador é energizado e fica apto a executar o programa. Os cilindros pneumáticos A e B estão recuados (a0 e b0 são verdadeiros) e o cilindro C está avançado (c1 é verdadeiro).

Programa P1

Transição inicial: Inicio → "EspC1" Lógica: "Partida".

O estado "EspC1" é ativado.

Estado: "EspC1"

A variável interna "X" é feita falsa. Esta variável foi criada para controlar o avanço do cilindro A. Quando "X" é falso é colocado o primeiro conjunto de 9 pacotes no fundo da caixa. Quando "X" é verdadeiro é colocado o segundo conjunto de 9 pacotes no início da caixa.

Transição: "EspC1" → "SobeCaixa" Lógica: "p"

O operador da máquina colocou uma caixa vazia no suporte do cilindro C e pressionou o botão "p". O estado "EspC1" é desativado e o estado "SobeCaixa" é ativado.

Estado: “SobeCaixa”

A saída digital “C+” é ativada para subir a caixa de papelão desde os roletes até a posição de colocação dos 18 pacotes de produtos.

Transição: “SobeCaixa” → “EspC2” Lógica: “c0”

A caixa de papelão está na posição correta e o cilindro C chegou ao final do curso (c0 é verdadeiro). O estado “SobeCaixa” é desativado e o estado “EspC2” é ativado.

Estado: “EspC2”

Nenhuma ação realizada. Quando este estado estiver ativo (verdadeiro) será usado como transição no programa P2.

Transição: “EspC2” → “DesceCaixa” Lógica: “EspA2.$\overline{X}$”

Todos os 18 pacotes de produto estão dentro da caixa de papelão. O estado “EspC2” é desativado e o estado “DesceCaixa” é ativado.

Nota. Observe que a variável X é falsa e o estado EspA2 é falso, então a transição “EspA2.$\overline{X}$” é falsa. Quando da colocação do primeiro conjunto de 9 pacotes na caixa de papelão, “X” é feito verdadeiro (X = $\overline{X}$) no estado “RecuaA”, e quando o estado “EspA2” estiver ativo (verdadeiro) a transição “EspA2.$\overline{X}$” é falsa. Quando da colocação do segundo conjunto de 9 pacotes na caixa de papelão, “X” é feito falso (X = $\overline{X}$) no estado “RecuaA”, e quando o estado “EspA2” estiver ativo (verdadeiro) a transição “EspA2.$\overline{X}$” é verdadeira.

Estado: “DesceCaixa”

A saída digital “C-” é ativada para descer a caixa de papelão desde a posição de colocação dos 18 pacotes de produtos até os roletes.

Transição: “DesceCaixa” → “EspC1” Lógica: “c1”

A caixa de papelão está sobre os roletes e o cilindro C chegou ao final do curso (c1 é verdadeiro). O estado “DesceCaixa” é desativado e o estado “EspC1” é ativado.

Programa P2

Transição inicial: Inicio → “EspB1” Lógica: “Partida”.

O estado “EspB1” é ativado.

Estado: “EspB1”

Nenhuma ação realizada.

Transição: “EspB1” → “SobeB” Lógica: “EspC2.t1”

A caixa de papelão está na posição correta (EspC2 é verdadeiro) e existem 3 pacotes de produtos na mesa do cilindro B (t1 é verdadeiro). O estado “EspB1” é desativado e o estado “SobeB” é ativado.

Estado: “SobeB”

A saída digital “B+” é ativada para subir os 3 pacotes de produto até a trava de pacotes.

Transição: “SobeB” → “DesceB” Lógica: “b1.$\overline{t2}$”

O embolo do cilindro B chegou ao final de curso (b1 é verdadeiro) e ainda não tem os 9 pacotes de produtos na mesa do cilindro B (t2 é falso). O estado “SobeB” é desativado e o estado “DesceB” é ativado.

Transição: “SobeB” → “EspB2” Lógica: “b1.t2”

O embolo do cilindro B chegou ao final de curso (b1 é verdadeiro) e tem os 9 pacotes de produtos na mesa do cilindro B (t2 é verdadeiro).O estado “SobeB” é desativado e o estado “EspB2” é ativado.

Estado: “EspB2”

Nenhuma ação realizada. Quando este estado estiver ativo (verdadeiro) será usado como transição no programa P3.

Transição: “EspB2” → “DesceB” Lógica: “EspA2”

Foi colocado 9 pacotes de produto dentro da caixa de papelão (EspA2 é verdadeiro). O estado “EspB2” é desativado e o estado “DesceB” é ativado.

Estado: “DesceB”

A saída digital “B-” é ativada para descer a mesa de acumulação de 3 pacotes.

Transição: “DesceB” → “EspB1” Lógica: “b0”

O embolo do cilindro B chegou ao final de curso (b0 é verdadeiro). O estado “DesceB” é desativado e o estado “EspB1” é ativado.

Programa P3

Transição inicial: Inicio → “EspA1” Lógica: “Partida”.

O estado “EspA1” é ativado.

Estado: “EspA1”

Nenhuma ação realizada.

Transição: “EspA1” → “EmpurraA” Lógica: “EspB2”

Os 9 pacotes de produto estão prontos para serem transferidos para a caixa de papelão (EspB2 é verdadeiro). O estado “EspA1” é desativado e o estado “EmpurraA” é ativado.

Estado: “EmpurraA”

A saída digital “A+” é ativada para empurrar os 9 pacotes de produto até a caixa de papelão.

Transição: “EmpurraA” → “RecuaA” Lógica: “$a2.\overline{X}+a1.X$”

O embolo do cilindro A chegou ao final de curso (a2 ou a1 é verdadeiro) e depende da condição da variável interna “X”. O estado “EmpurraA” é desativado e o estado “RecuaA” é ativado.

Nota. A variável interna “X” começa sendo falsa, então a transição “$a2.\overline{X}$” é verdadeiro e o embolo do cilindro A empurra os 9 pacotes até o fundo da caixa de papelão (a2 é verdadeiro). No estado “RecuaA” a variável “X” é feita verdadeira ($X=\overline{X}$) e na próxima execução a transição “a1.X” é verdadeira e o embolo do cilindro A empurra os 9 pacotes até o início da caixa de papelão (a1 é verdadeiro).

Estado: “RecuaA”

A saída digital “A-” é ativada para recuar o embolo do cilindro A e a variável “X” tem a lógica invertida.

Transição: “RecuaA” → “EspA2” Lógica: “a0”

O embolo do cilindro A chegou ao final de curso (a0 é verdadeiro). O estado “RecuaA” é desativado e o estado “EspA2” é ativado.

Estado: “EspA2”

Nenhuma ação realizada.

Transição: “EspA2” → “EspA1” Lógica: “EspB1”

O programa P2 está no início com o embolo do cilindro B totalmente recuado. O estado “EspA2” é desativado e o estado “EspA1” é ativado.

Codificação do diagrama de estados em linguagem Ladder

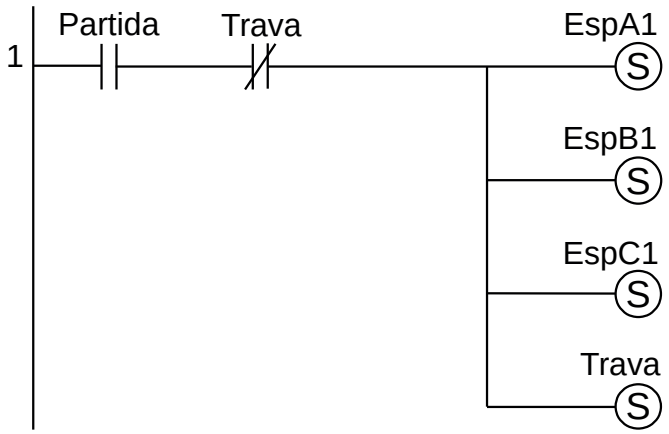

Figura 7.38 – Programa em linguagem Ladder para a inicialização.

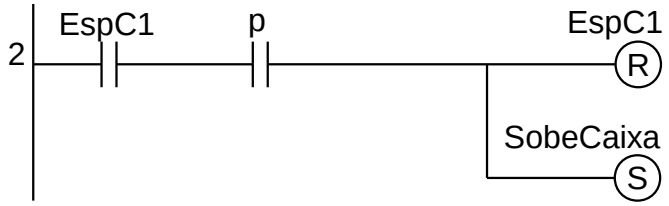

Figura 7.39 – Programa em linguagem Ladder para as transições saintes do estado EspC1.

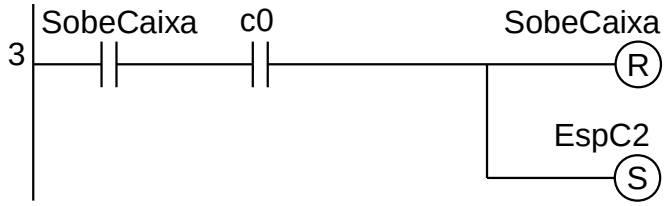

Figura 7.40 – Programa em linguagem Ladder para as transições saintes do estado SobeCaixa.

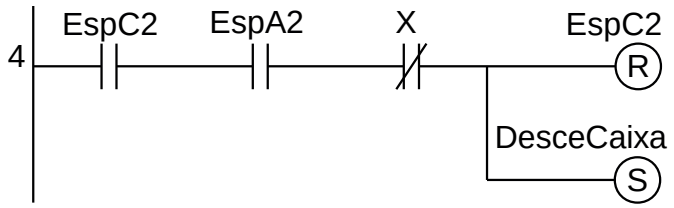

Figura 7.41 – Programa em linguagem Ladder para as transições saintes do estado EspC2.

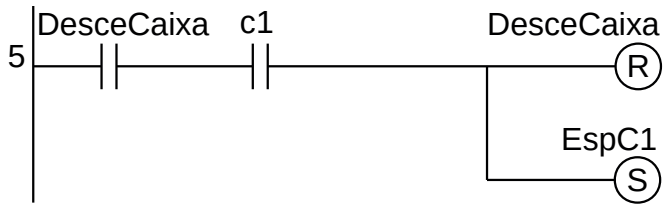

Figura 7.42 – Programa em linguagem Ladder para as transições saintes do estado DesceCaixa.

Figura 7.43 – Programa em linguagem Ladder para as transições saintes do estado EspB1.

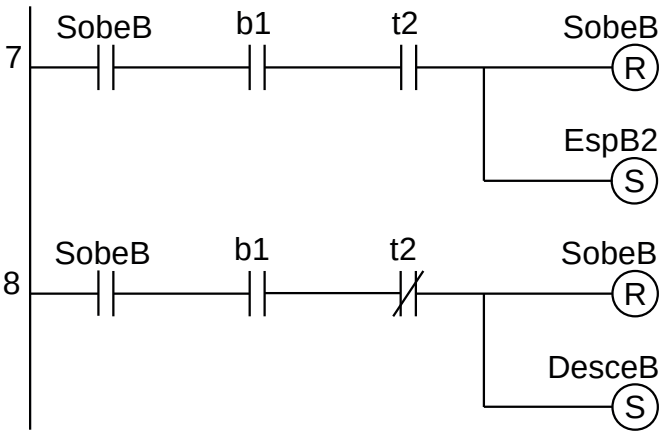

Figura 7.44 – Programa em linguagem Ladder para as transições saintes do estado SobeB.

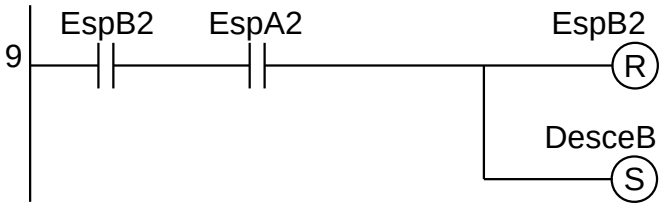

Figura 7.45 – Programa em linguagem Ladder para as transições saintes do estado EspB2.

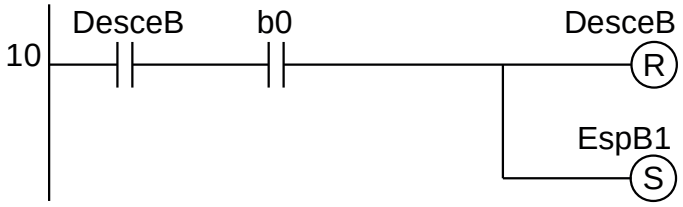

Figura 7.46 – Programa em linguagem Ladder para as transições saintes do estado DesceB.

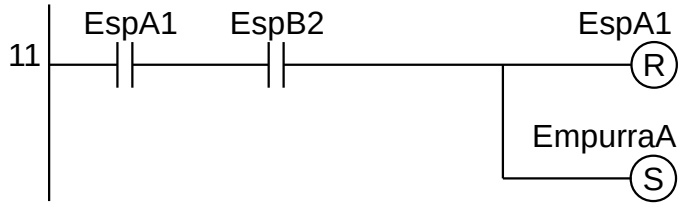

Figura 7.47 – Programa em linguagem Ladder para as transições saintes do estado EspA1.

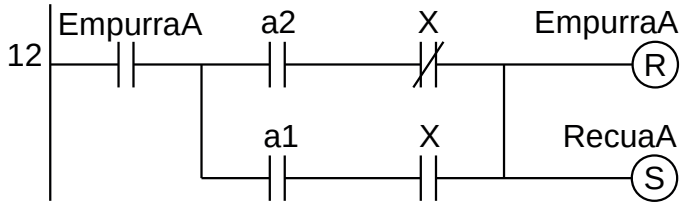

Figura 7.48 – Programa em linguagem Ladder para as transições saintes do estado EmpurraA.

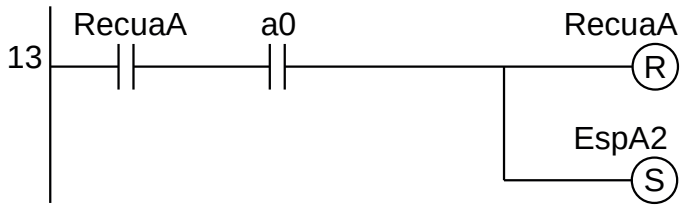

Figura 7.49 – Programa em linguagem Ladder para as transições saintes do estado RecuaA.

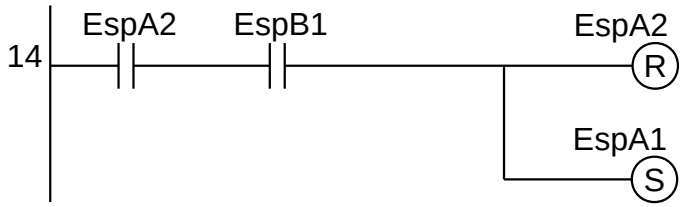

Figura 7.50 – Programa em linguagem Ladder para as transições saintes do estado EspA2.

Para a ativação das ações sobre a variável interna “X” pode-se observar que o modo de operação desta variável é semelhante ao problema do botão e da lâmpada. Assim o mesmo programa de controle do botão e da lâmpada foi utilizado para executar a mudança de lógica na variável interna “X”, acrescentando a ação do estado “EspC1”. Esse programa pode ser visto na figura 7.51.

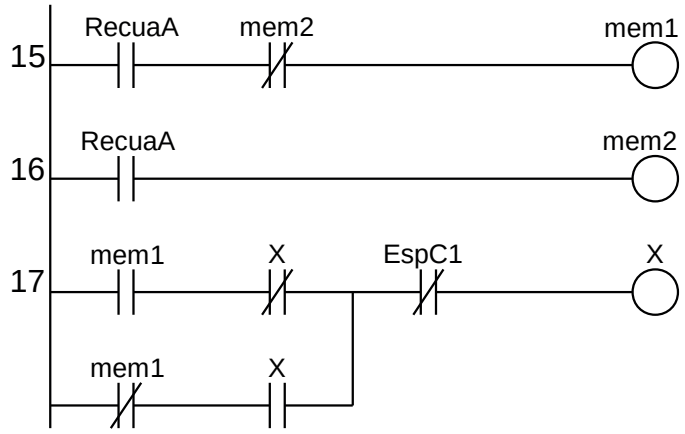

Figura 7.51 – Programa em linguagem Ladder para variáveis internas.

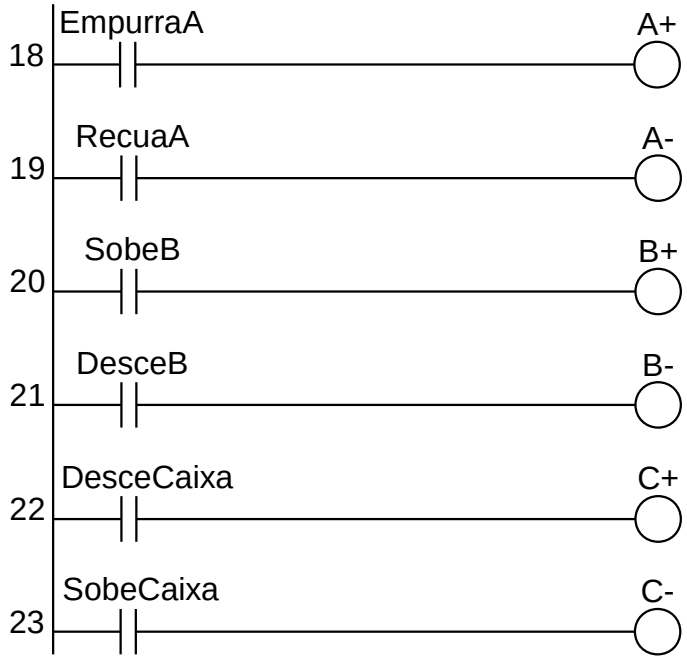

Figura 7.52 – Programa em linguagem Ladder para ativação das saídas.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 7.7 e 7.8. A tabela com as variáveis internas pode ser vista na tabela 7.9.

Variável	Etiqueta	Descrição
M01	EspC1	Memória para estado.
M02	SobeCaixa	Memória para estado.
M03	EspC2	Memória para estado.
M04	DesceCaixa	Memória para estado.
M05	EspB1	Memória para estado.
M06	SobeB	Memória para estado.
M07	EspB2	Memória para estado.
M08	DesceB	Memória para estado.
M09	EspA1	Memória para estado.
M10	EmpurraA	Memória para estado.
M11	RecuaA	Memória para estado.
M12	EspA2	Memória para estado.
M13	Trava	Memória para estado.
M14	mem1	Memória para estado.
M15	mem2	Memória para estado.
M16	X	Memória para estado.

Tabela 7.9 – Descrição das variáveis internas.

7.5.4 – Exemplo 4. Máquina de engarrafamento

Esse exemplo mostra uma máquina composta por estágios que podem funcionar em simultâneo. Um dos estágios ativa os acionadores somente se os outros estágios estiverem parados.

Descrição da máquina

Uma máquina automática de enchimento de líquido e colocação da tampa é mostrada na figura 7.53. Esta máquina é composta de 3 estações de atividades sendo a estação1 para transferência do frasco da esteira acumuladora para a esteira de compasso, a estação 2 para dosagem do líquido e enchimento do frasco e a estação 3 para colocação e rosqueamento da tampa.

Estação 1: carga ou transferência

- O sensor “CP1” identifica garrafa vazia pronta para ser transferida.
- O cilindro A empurra a garrafa da esteira livre de entrada para a esteira de compasso para ser enchida.
- Os sensores “a0” e “a1”, do tipo fim-de-curso, identificam a posição de início e final do embolo do cilindro A.
- O acionamento do cilindro A é feito através de válvula pneumática controlada pelos solenoides “a+” e “a-” que tem a função de avanço e recuo do embolo do cilindro.
- O cilindro B avança a esteira de compasso para nova posição, através de um sistema de pinhão, cremalheira e catraca de simples avanço e retorno livre.
- Os sensores “b0” e “b1”, do tipo fim-de-curso, identificam a posição de início e final do embolo do cilindro B
- O acionamento do cilindro B é feito através de válvula pneumática controlada pelos solenoides “b+” e “b-” que tem a função de avanço e recuo do embolo do cilindro.

Estação 2: dosagem e enchimento.

- O sensor “CP2” identifica garrafa vazia debaixo do bocal de enchimento de líquido.
- O cilindro C recua para fazer a dosagem de líquido e empurra para colocar o líquido na garrafa.
- Os sensores “c0” e “c1”, do tipo fim-de-curso, identificam a posição de início e final do embolo do cilindro C.
- O acionamento do cilindro C é feito através de válvula pneumática controlada pelos solenoides “c+” e “c-” que tem a função de avanço e recuo do embolo do cilindro.
- A válvula D é acionada pelo solenoide “d+” e tem a função de liberar o líquido para a garrafa.

Estação 3: colocação da tampa

- O sensor “CP3” identifica garrafa cheia debaixo do sistema de rosqueamento.
- O cilindro G empurra a tampa para debaixo do sistema de rosqueamento. A tampa fica presa num anel de borracha do rosqueador quando o cilindro E descer.
- Os sensores “g0” e “g1”, do tipo fim-de-curso, identificam a posição de início e final do embolo do cilindro G.
- O acionamento do cilindro G é feito através de válvula pneumática controlada pelos solenoides “g+” e “g-” que tem a função de avanço e recuo do embolo do cilindro.
- O cilindro E captura a tampa (primeiro movimento) e empurra a tampa para a ponta da garrafa (segundo movimento).
- O sensor “e1”, do tipo pressostato, identifica o momento que o cilindro E encontra algum obstáculo.
- O sensor “e0”, do tipo fim-de-curso, identifica a posição de início do embolo do cilindro E.

- O acionamento do cilindro E é feito através de válvula pneumática controlada pelos solenoides “e+” e “e-” que tem a função de avanço e recuo do embolo do cilindro.
- A válvula F é acionada pelo solenoide “f+” e tem a função de acionar o motor pneumático que rosqueia a tampa. O final de rosqueamento é feito após 5 segundos. A tampa é liberada por um sistema de segurança tipo mola catraca.
- Existe um painel com uma botoeira para início de operação com etiqueta “Partida” e uma botoeira para término de operação com etiqueta “Parada”.

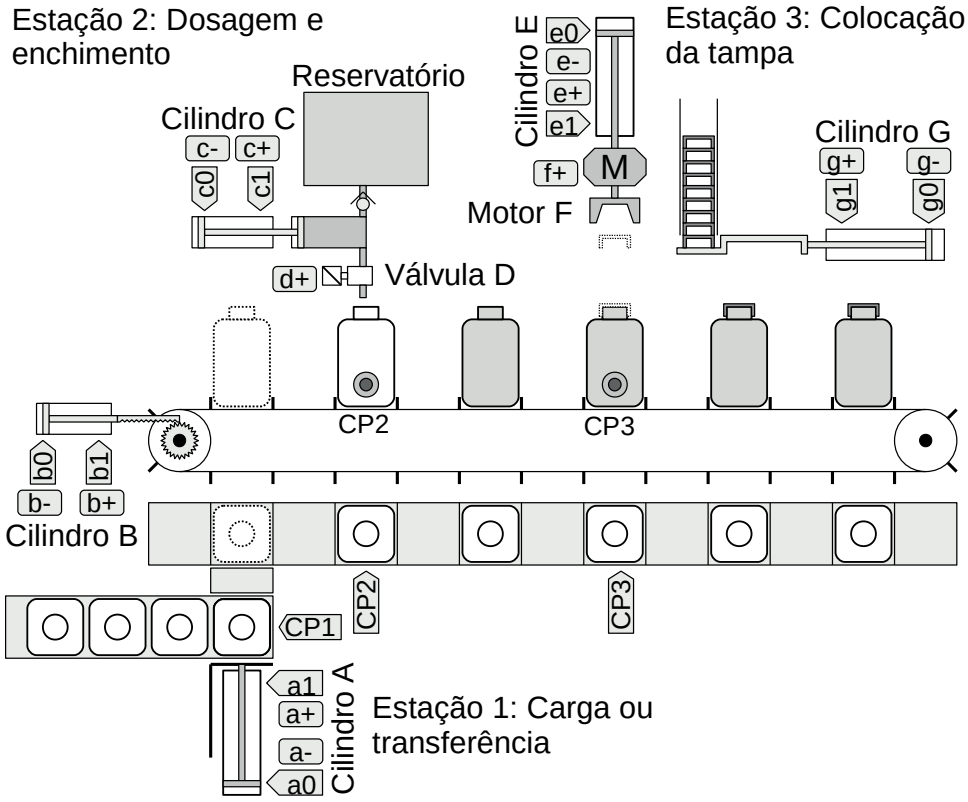

Figura 7.53 – Máquina de engarrafamento automático.

Descrição do funcionamento

Condições iniciais

As condições seguintes são verificadas pelo operador da máquina antes da colocação em funcionamento automático.

- Todos os cilindros pneumáticos recuados.
- Nenhuma garrafa na esteira de compasso.
- Sem tampa no dispositivo de rosqueamento.
- Reservatório de líquido cheio.
- Alimentador de tampas cheio.

O programa de funcionamento automático não precisa fazer a verificação ou colocar a máquina nestas condições.

Funcionamento automático normal

O processo de engarrafamento deve funcionar normalmente para as seguintes condições:

1. Apenas uma garrafa.

2. Sem falta de garrafas.
3. Falta momentânea de garrafa.
4. Falta de todas as garrafas.

Para todas as condições, nenhuma garrafa deve ficar sobre a esteira de compasso, ou seja, a esteira deve ser esvaziada.

O processo deve ser iniciado quando o operador da máquina pressionar a botoeira com etiqueta "Partida". Quando o operador da máquina pressionar a botoeira com etiqueta "Parada" todas as estações devem terminar os processos. A esteira de compasso não deve ser esvaziada.

Estação 1: carga ou transferência

Esta estação somente pode funcionar quando as outras 2 estações estiverem paradas em simultâneo.

Esta estação possui dois modos operacionais:

Modo 1 – Transferência de um frasco da esteira acumuladora para a esteira de compasso.

- O sensor CP1 é verdadeiro.
- O cilindro A empurra o frasco para a esteira de compasso.
- Um contador regressivo é carregado com valor 6.
- O cilindro B avança a esteira de compasso.
- O contador é decrementado.

Modo 2 – Esvaziamento da esteira de compasso.

- O sensor CP1 é falso e o contador não é zero.
- O cilindro B avança a esteira de compasso.
- O contador é decrementado.

Estação 2: dosagem e enchimento

Esta estação somente pode funcionar se o sensor de frascos CP2 passar de falso para verdadeiro.

Esta estação possui um único modo operacional.

- A válvula solenoide D é aberta e o cilindro C avança para encher o frasco.
- A válvula solenoide D é fechada e o cilindro C recua para encher o dosador.

Estação 3: colocação da tampa

Esta estação somente pode funcionar se o sensor de frascos CP3 passar de falso para verdadeiro.

Esta estação possui dois modos operacionais:

Modo 1 – Captura da tampa.

- O cilindro G avança para posicionar a tampa para captura.
- O cilindro E avança para "pegar" a tampa.
- O cilindro E recua.
- O cilindro G recua.

Modo 2 – Rosqueamento.

- O cilindro E avança com a tampa até a boca do frasco.
- O motor pneumático F gira por 5 segundos.
- O cilindro E recua.

Os dois modos operacionais devem ser executados na ordem modo 1 e depois modo 2.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entradas é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 7.10.

Parafuso	Etiqueta	Descrição
I01	Partida	Botoeira NA.
I02	Parada	Botoeira NA.
I03	CP1	Sensor capacitivo de presença.
I04	CP2	Sensor capacitivo de presença.
I05	CP3	Sensor capacitivo de presença.
I06	a0	Sensor magnético tipo "reed".
I07	a1	Sensor magnético tipo "reed".
I08	b0	Sensor magnético tipo "reed".
I09	b1	Sensor magnético tipo "reed".
I10	c0	Sensor magnético tipo "reed".
I11	c1	Pressostato.
I12	e0	Sensor magnético tipo "reed".
I13	e1	Sensor magnético tipo "reed".
I14	g0	Sensor magnético tipo "reed".
I15	g1	Sensor magnético tipo "reed".

Tabela 7.10 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 7.11.

Parafuso	Etiqueta	Descrição
Q01	a+	Solenoide de eletroválvula pneumática.
Q02	a-	Solenoide de eletroválvula pneumática.
Q03	b+	Solenoide de eletroválvula pneumática.
Q04	b-	Solenoide de eletroválvula pneumática.
Q05	c+	Solenoide de eletroválvula pneumática.
Q06	c-	Solenoide de eletroválvula pneumática.
Q07	d+	Solenoide de eletroválvula de líquido.
Q08	e+	Solenoide de eletroválvula pneumática.
Q09	e-	Solenoide de eletroválvula pneumática.
Q10	f+	Solenoide de eletroválvula pneumática.
Q11	g+	Solenoide de eletroválvula pneumática.
Q12	g-	Solenoide de eletroválvula pneumática.

Tabela 7.11 – Descrição das saídas.

Parte 1 – Inicialização

É criado a variável “Func” que indica que o programa do controlador pode ser executado.

Esta variável é controlada pelas botoeiras “Partida” e “Parada”. Também é criada a variável “Trava” para inibir o reinício do primeiro bloco do diagrama de estados através da botoeira de inicialização.

É utilizado frases lógicas para a modelagem dessa etapa.

A primeira frase produz um “pulso único” da botoeira “Partida”. Esse pulso ativa os primeiros blocos dos diagramas de estados de cada estação.

SE (“Partida” é verdadeiro e “Trava” é falso) ENTÃO (fazer “Espera1” verdadeiro e memorizar e fazer “Espera2” verdadeiro e memorizar e fazer “Espera3” verdadeiro e memorizar e fazer “Trava” verdadeiro e memorizar)

A segunda frase permite o reinício do programa, habilitando o funcionamento do programa.

SE (“Partida” é verdadeiro) ENTÃO (fazer a variável “Func” verdadeiro e memorizar).

A terceira frase desabilita a variável de funcionamento do programa.

SE (“Parada” é verdadeiro) ENTÃO (fazer a variável “Func” falso).

Parte 2 – Diagrama de estados

O diagrama de estados para controlar o funcionamento das 3 estações da máquina de engarrafamento automático pode ser visto na figura 7.54.

Condição inicial

Considere que todos os cilindros pneumáticos estão recuados, todas as eletroválvulas estão desativadas, nenhuma garrafa sobre as esteiras de entrada e de compasso, ou seja, CP1 falso, CP2 falso e CP3 falso. Os sensores de embolo de cilindro recuados, a0, b0, c0, e0 e g0, estão todos em verdadeiro. Os sensores de embolo de cilindro avançados, a1, b1, c1, e1 e g1, estão todos em falso.

Estação 1: carga ou transferência

Estado inicial: energização.

O controlador é energizado e fica apto a executar o programa.

Transição inicial: inicio → “Espera1” Lógica: “Partida”.

O estado “Espera1” é ativado.

Estado: “Espera1”

Nenhuma ação é realizada, o controlador fica aguardando a condição ideal para iniciar o processo de transferência de frascos.

Frases lógicas são montadas para facilitar a transição do estado “Espera1” para outros estados.

A frase lógica para “Log0” considera que a estação de enchimento e estação de colocar a tampa estão paradas. Essas estações podem estar paradas quando não tem tarefa a realizar (Espera2 e Espera 3) ou a tarefa já foi realizada (Espera2a e Espera3a).

> SE ((“Espera2” é verdadeiro ou “Espera2a” é verdadeiro) e (“Espera3” é verdadeiro ou “Espera3a” é verdadeiro)) ENTÃO (fazer a variável “Log0” verdadeiro).

A frase lógica para “Log1” considera a condição de “Log0” é verdadeiro e a existência de um frasco na esteira acumuladora (“CP1” é verdadeiro) pronto para ser transferido para a esteira de compasso.

> SE (“Log0” é verdadeiro e “CP1” é verdadeiro) ENTÃO (fazer a variável “Log1” verdadeiro).

A frase lógica para “Log2” considera a condição de “Log0” é verdadeiro e a não existência de um frasco na esteira acumuladora (“CP1” é falso) e o contador de frascos na esteira de compasso é diferente de 0 (“Cont” é falso).

> SE (“Log0” é verdadeiro e “CP1” é falso e “Cont” é falso) ENTÃO (fazer a variável “Log2” verdadeiro).

Transição: “Espera1” → “Avança_A” Lógica: “Log1.Func”

As estações de enchimento e colocar a tampa estão paradas e existe uma garrafa na esteira de acumulação pronta para ser transferida para a esteira de compasso, “Log1” é verdadeiro. A variável “Func” deve estar ativa (verdadeira). O estado “Espera1” é desativado e o estado “Avança_A” é ativado.

Transição: “Espera1” → “Avança_B” Lógica: “Log2.Func”

As estações de enchimento e colocar a tampa estão paradas e não exite uma garrafa na esteira de acumulação e o contador é diferente de zero, “Log2” é verdadeiro. A variável “Func” deve estar ativa (verdadeira). O estado “Espera1” é desativado e o estado “Avança_B” é ativado.

Estado: “Avança_A”

A eletroválvula “a+” é ativada para permitir o embolo do cilindro A empurrar a garrafa da esteira de acumulação para a esteira de compasso. O contador “Cont” é iniciado, ou seja, o registrador interno de contagem assume o valor pré-determinado (valor=6).

Transição: “Avança_A” → “Recua_A” Lógica: “(a1)”

O embolo do cilindro A chegou a final do curso e o sensor “a1” torna-se verdadeiro.

O estado “Avança_A” é desativado e o estado “Recua_A” é ativado.

Estado: “Recua_A”

A eletroválvula “a-” é ativada para permitir o embolo do cilindro A recuar e ficar pronto para empurrar outro frasco.

Transição: “Recua_A” → “Avança_B” Lógica: “(a0)”

O embolo do cilindro A chegou ao início do curso e o sensor “a0” torna-se verdadeiro.

O estado “Recua_A” é desativado e o estado “Avança_B” é ativado.

Estado: “Avança_B”

A eletroválvula “b+” é ativada para permitir o embolo do cilindro B mover a esteira de compasso em uma unidade de movimento. Um sinal de contagem regressiva é enviado para o contador “Cont”.

Transição: “Avança_B” → “Recua_B” Lógica: “(b1)”

O embolo do cilindro B chegou a final do curso e o sensor “b1” torna-se verdadeiro.

O estado “Avança_B” é desativado e o estado “Recua_B” é ativado.

Estado: “Recua_B”

A eletroválvula “b-” é ativada para permitir o embolo do cilindro B recuar e ficar pronto para mover novamente a esteira de compasso.

Transição: “Recua_B” → “Espera1” Lógica: “(b0)”

O embolo do cilindro B chegou ao início do curso e o sensor “b0” torna-se verdadeiro.

O estado “Recua_B” é desativado e o estado “Espera1” é ativado.

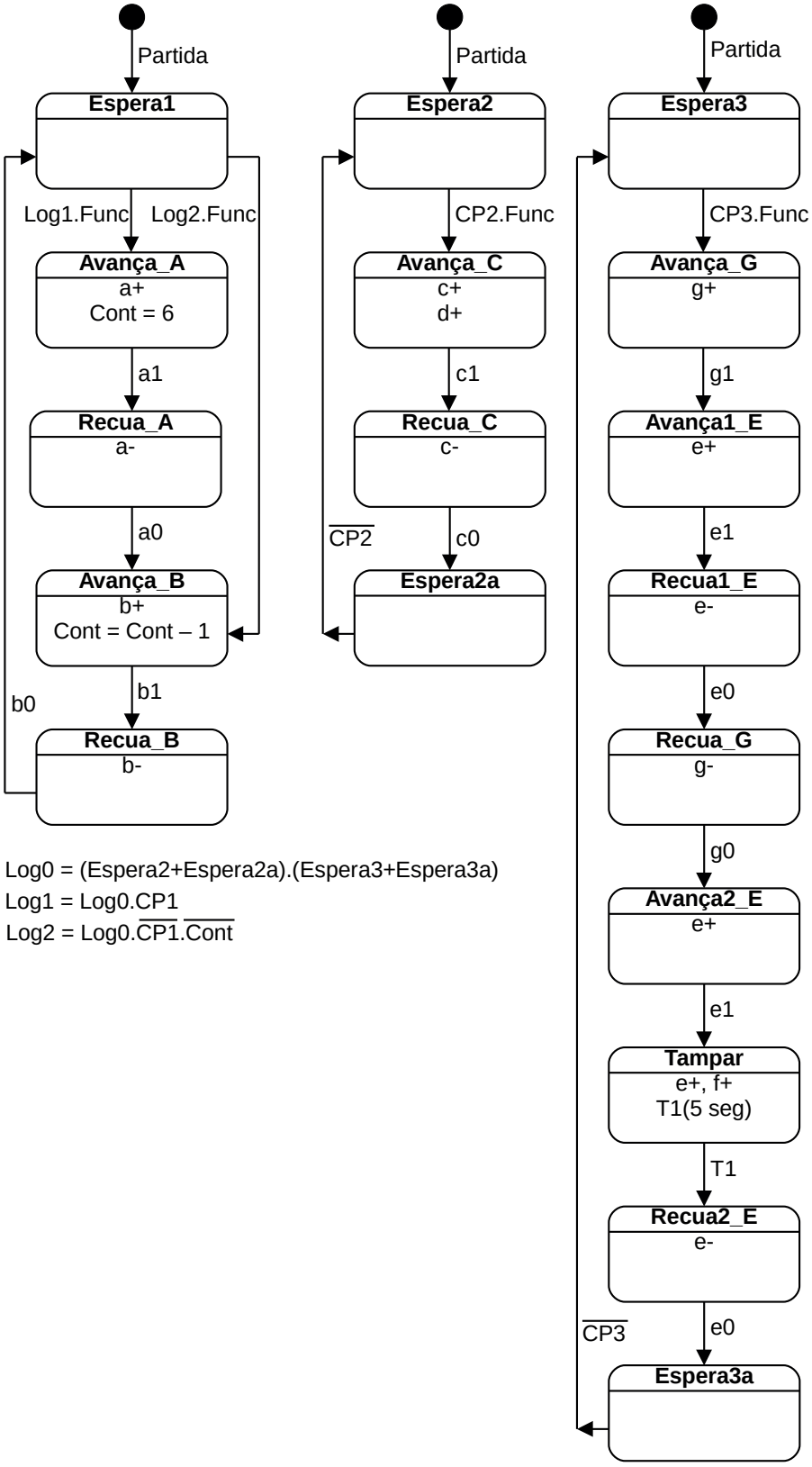

Figura 7.54 – Diagrama de estados para o controle da máquina de engarrafamento.

Estação 2: enchimento

Estado inicial: energização.

O controlador é energizado e fica apto a executar o programa.

Transição inicial: inicio → “Espera2” Lógica: “Partida”.

O estado “Espera2” é ativado.

Estado: “Espera2”

Nenhuma ação é realizada, o controlador fica aguardando um frasco vazio se posicionar para ser enchido.

Transição: “Espera2” → “Avança_C” Lógica: “CP2.Func”

Um frasco vazio se posicionou para ser enchido. A variável “Func” deve estar ativa (verdadeira). O estado “Espera2” é desativado e o estado “Avança_C” é ativado.

Estado: “Avança_C”

A eletroválvula “c+” é ativada para permitir o embolo do cilindro C empurrar o líquido para dentro do frasco vazio. A eletroválvula “d+” é ativada para abrir a válvula de líquido para encher o frasco vazio.

Transição: “Avança_C” → “Recua_C” Lógica: “(c1)”

O embolo do cilindro C chegou a final do curso e o sensor “c1” torna-se verdadeiro.

O estado “Avança_C” é desativado e o estado “Recua_C” é ativado.

Estado: “Recua_C”

A eletroválvula “c-” é ativada para permitir o embolo do cilindro C recuar e encher o compartimento de dosagem de líquido.

Transição: “Recua_C” → “Espera2a” Lógica: “(c0)”

O embolo do cilindro C chegou ao início do curso e o sensor “c0” torna-se verdadeiro.

O estado “Recua_C” é desativado e o estado “Espera2a” é ativado.

Estado: “Espera2a”

Nenhuma ação é realizada, o controlador fica aguardando um frasco cheio ser deslocado pela esteira de compasso.

Transição: “Espera2a” → “Espera2” Lógica: “($\overline{CP2}$)”

O frasco cheio foi deslocado e nenhum frasco é detectado pelo sensor “CP2”.

O estado “Espera2a” é desativado e o estado “Espera2” é ativado.

Estação 3: colocar a tampa

Estado inicial: energização.

O controlador é energizado e fica apto a executar o programa.

Transição inicial: inicio → “Espera3” Lógica: “Partida”.

O estado “Espera3” é ativado.

Estado: “Espera3”

Nenhuma ação é realizada, o controlador fica aguardando um frasco vazio se posicionar para ser tampado.

Transição: “Espera3” → “Avança_G” Lógica: “CP3.Func”

Um frasco vazio se posicionou para ser tampado. A variável “Func” deve estar ativa (verdadeira). O estado “Espera3” é desativado e o estado “Avança_G” é ativado.

Estado: “Avança_G”

A eletroválvula “g+” é ativada para permitir o embolo do cilindro G empurrar uma tampa para a posição do rosqueador.

Transição: “Avança_G” → “Avança1_E” Lógica: “(g1)”

O embolo do cilindro G chegou a final do curso e o sensor “g1” torna-se verdadeiro.

O estado “Avança_G” é desativado e o estado “Avança1_E” é ativado.

Estado: “Avança1_E”

A eletroválvula “e+” é ativada para permitir o embolo do cilindro E avançar até que a tampa fique presa no rosqueador.

Transição: “Avança1_E” → “Recua1_E” Lógica: “(e1)”

O embolo do cilindro E chegou a final do curso e o sensor “e1” torna-se verdadeiro.

O estado “Avança1_E” é desativado e o estado “Recua1_E” é ativado.

Estado: “Recua1_E”

A eletroválvula “e-” é ativada para permitir o embolo do cilindro E recuar até o alto com a tampa presa no rosqueador.

Transição: “Recua1_E” → “Recua_G” Lógica: “(e0)”

O embolo do cilindro E chegou ao início do curso e o sensor “e0” torna-se verdadeiro.

O estado “Recua1_E” é desativado e o estado “Recua_G” é ativado.

Estado: “Recua_G”

A eletroválvula “g-” é ativada para permitir o embolo do cilindro G recuar e capturar uma tampa no alimentador de tampas.

Observe que o recuo do embolo do cilindro G pode ser feito após o embolo do cilindro E começar a se movimentar, neste caso deve ser analisado quando o sensor “e1” é falso.

Transição: “Recua_G” → “Avança2_E” Lógica: “(g0)”

O embolo do cilindro G chegou ao início do curso e o sensor “g0” torna-se verdadeiro.

O estado “Recua_G” é desativado e o estado “Avança2_E” é ativado.

Estado: “Avança2_E”

A eletroválvula “e+” é ativada para permitir o embolo do cilindro E avançar até que a tampa entre em contato com o frasco para ser rosqueada.

Transição: “Avança2_E” → “Tampar” Lógica: “(e1)”

O embolo do cilindro E chegou a final do curso e o sensor “e1” torna-se verdadeiro.

O estado “Avança2_E” é desativado e o estado “Tampar” é ativado.

Estado: “Tampar”

A eletroválvula “e+” é ativada para permitir o embolo do cilindro E fazer pressão sobre o rosqueador, garantindo o rosqueamento. A eletroválvula “f+” é ativada para permitir o giro do motor pneumático F. O temporizador T1, de 5 segundos, é ativado para limitar o tempo de rosqueamento.

Transição: “Tampar” → “Recua2_E” Lógica: “(T1)”

O temporizador T1, de 5 segundos, terminou a contagem.

O estado “Tampar” é desativado e o estado “Recua2_E” é ativado.

Transição: “Recua2_E” → “Espera3a” Lógica: “(e0)”

O embolo do cilindro E chegou ao início do curso e o sensor “e0” torna-se verdadeiro.

O estado “Recua2_E” é desativado e o estado “Espera3a” é ativado.

Estado: “Espera3a”

Nenhuma ação é realizada, o controlador fica aguardando um frasco tampado ser deslocado pela esteira de compasso.

Transição: “Espera3a” → “Espera3” Lógica: “($\overline{CP3}$)”

O frasco tampado foi deslocado e nenhum frasco é detectado pelo sensor “CP3”.

O estado “Espera3a” é desativado e o estado “Espera3” é ativado.

Codificação do diagrama de estados em linguagem Ladder

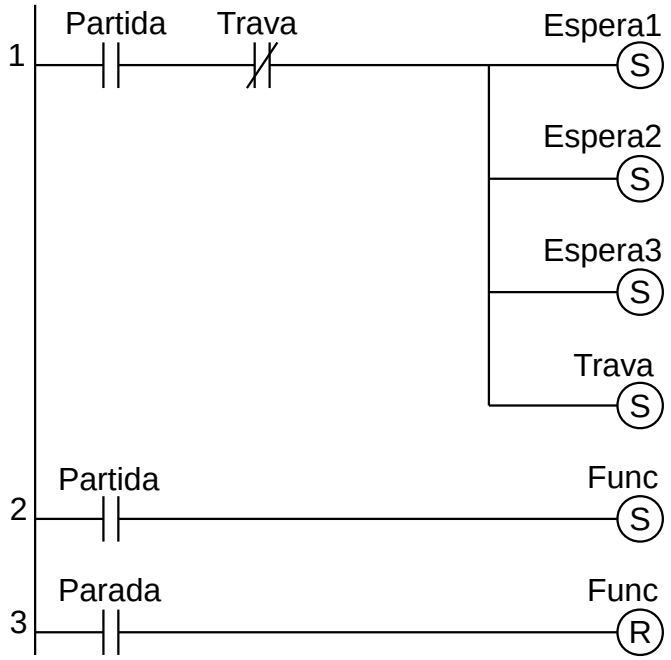

Figura 7.55 – Programa em linguagem Ladder para a inicialização.

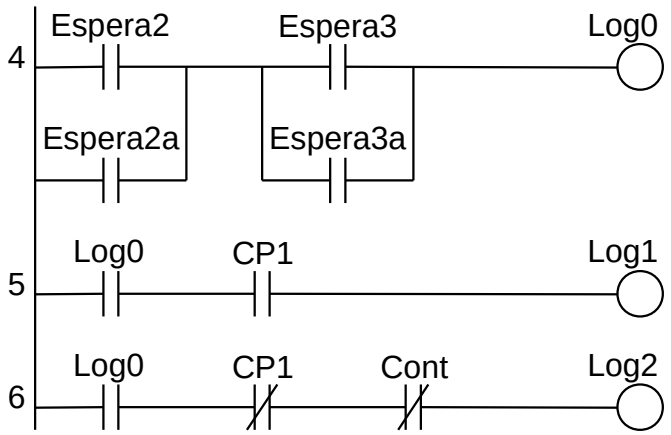

Figura 7.56 – Programa em linguagem Ladder para as frases lógicas Log0, Log1 e Log2.

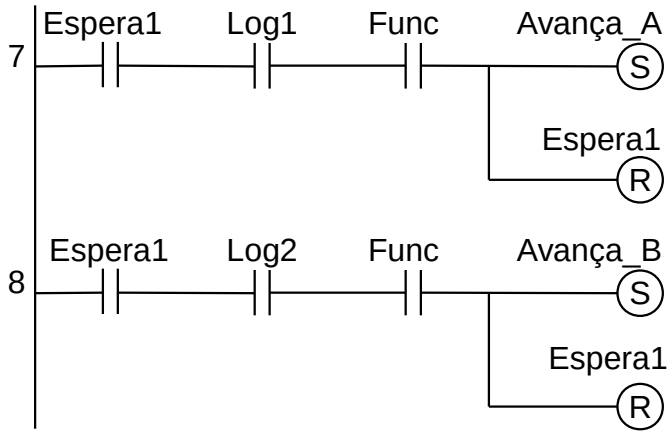

Figura 7.57 – Programa em linguagem Ladder para as transições saintes do estado Espera1.

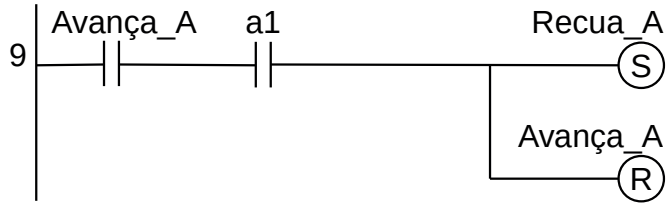

Figura 7.58 – Programa em linguagem Ladder para as transições saintes do estado Avança_A.

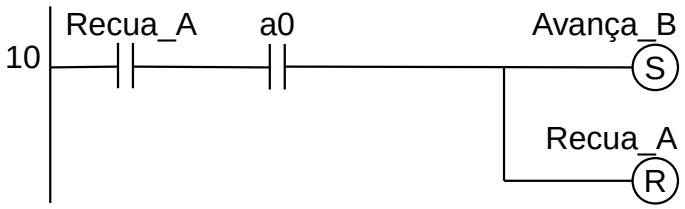

Figura 7.59 – Programa em linguagem Ladder para as transições saintes do estado Recua_A.

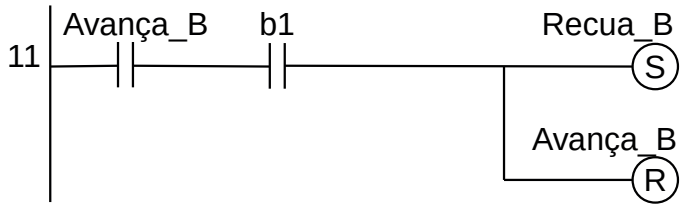

Figura 7.60 – Programa em linguagem Ladder para as transições saintes do estado Avança_B.

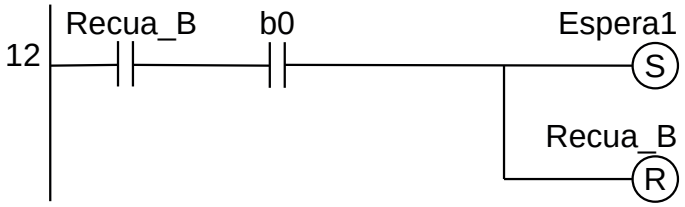

Figura 7.61 – Programa em linguagem Ladder para as transições saintes do estado Recua_B.

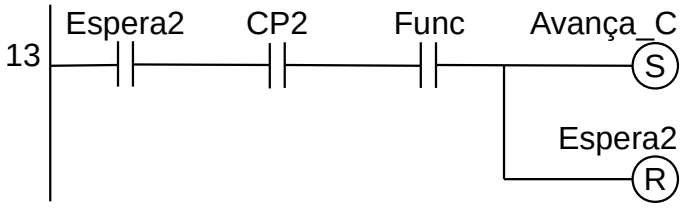

Figura 7.62 – Programa em linguagem Ladder para as transições saintes do estado Espera2.

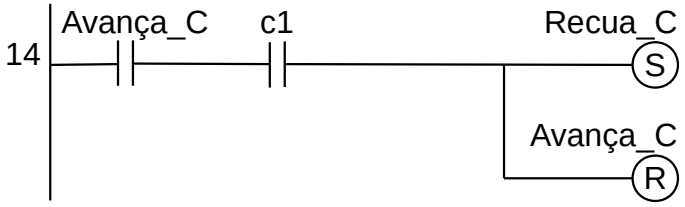

Figura 7.63 – Programa em linguagem Ladder para as transições saintes do estado Avança_C.

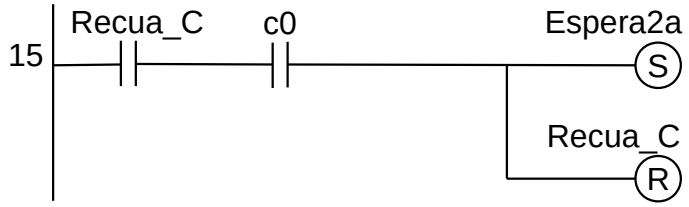

Figura 7.64 – Programa em linguagem Ladder para as transições saintes do estado Recua_C.

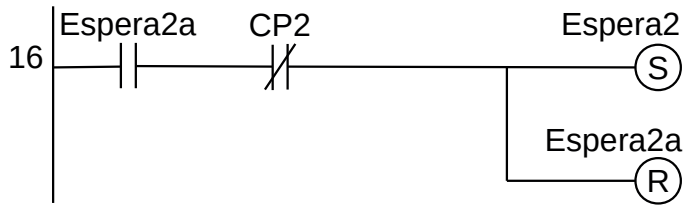

Figura 7.65 – Programa em linguagem Ladder para as transições saintes do estado Espera2a.

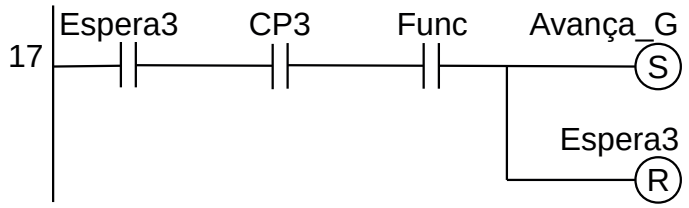

Figura 7.66 – Programa em linguagem Ladder para as transições saintes do estado Espera3.

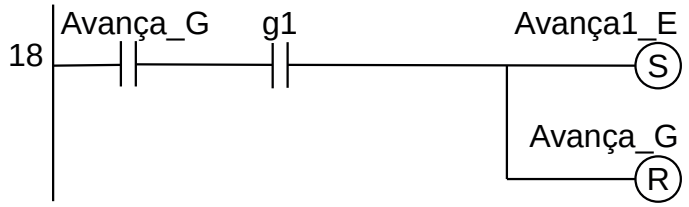

Figura 7.67 – Programa em linguagem Ladder para as transições saintes do estado Avança_G.

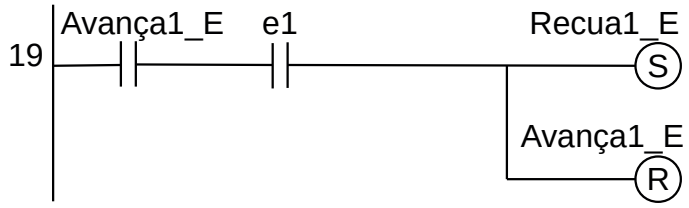

Figura 7.68 – Programa em linguagem Ladder para as transições saintes do estado Avança1_E.

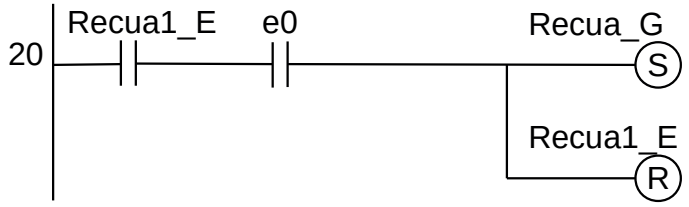

Figura 7.69 – Programa em linguagem Ladder para as transições saintes do estado Recua1_E.

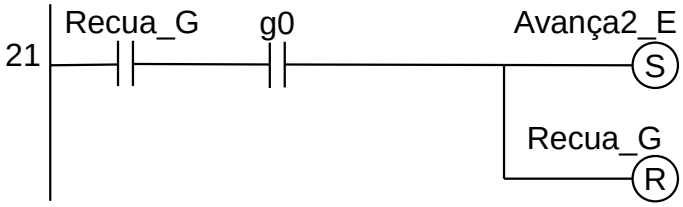

Figura 7.70 – Programa em linguagem Ladder para as transições saintes do estado Recua_G.

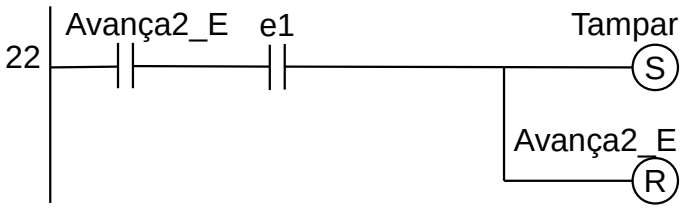

Figura 7.71 – Programa em linguagem Ladder para as transições saintes do estado Avança2_E.

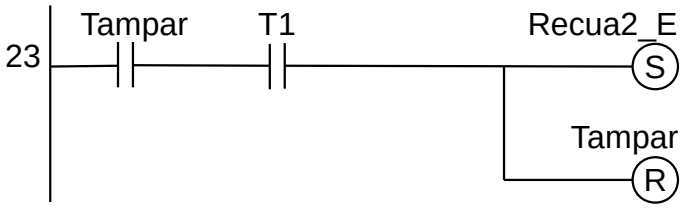

Figura 7.72 – Programa em linguagem Ladder para as transições saintes do estado Tampar.

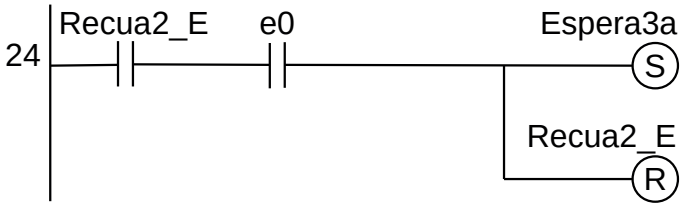

Figura 7.73 – Programa em linguagem Ladder para as transições saintes do estado Recua2_E.

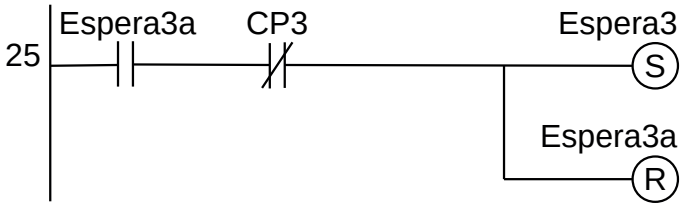

Figura 7.74 – Programa em linguagem Ladder para as transições saintes do estado Espera3a.

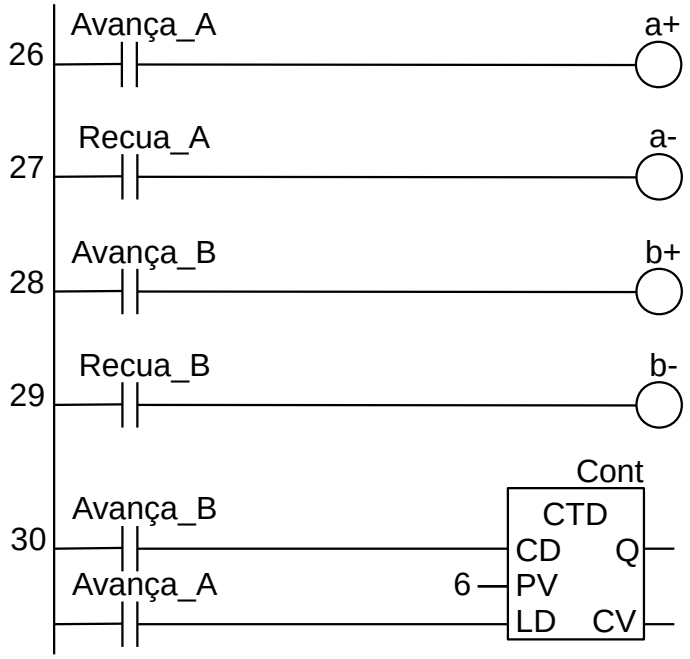

Figura 7.75 – Programa em linguagem Ladder para ativação das saídas da estação 1.

Observe que o símbolo do bloco do contador está de acordo com a norma.

- A entrada “CD” (do inglês “Counter Down”) é para executar a contagem regressiva.
- A entrada “PV” (do inglês “Preset Value”) é o valor de contagem a ser armazenado no registrador interno.
- A entrada “LD” (do inglês “Load”) é para iniciar o registrador interno do contador com o valor em “PV”.
- A saída “CV” (do inglês “Current Value”) é o valor atual do registrador interno.
- A saída “Q” é verdadeira quando o valor interno do registrador for igual a zero.

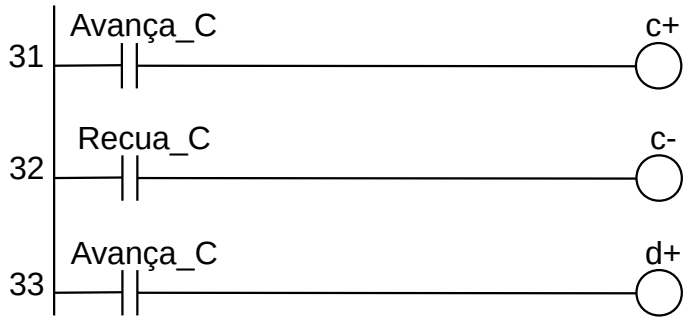

Figura 7.76 – Programa em linguagem Ladder para ativação das saídas da estação 2.

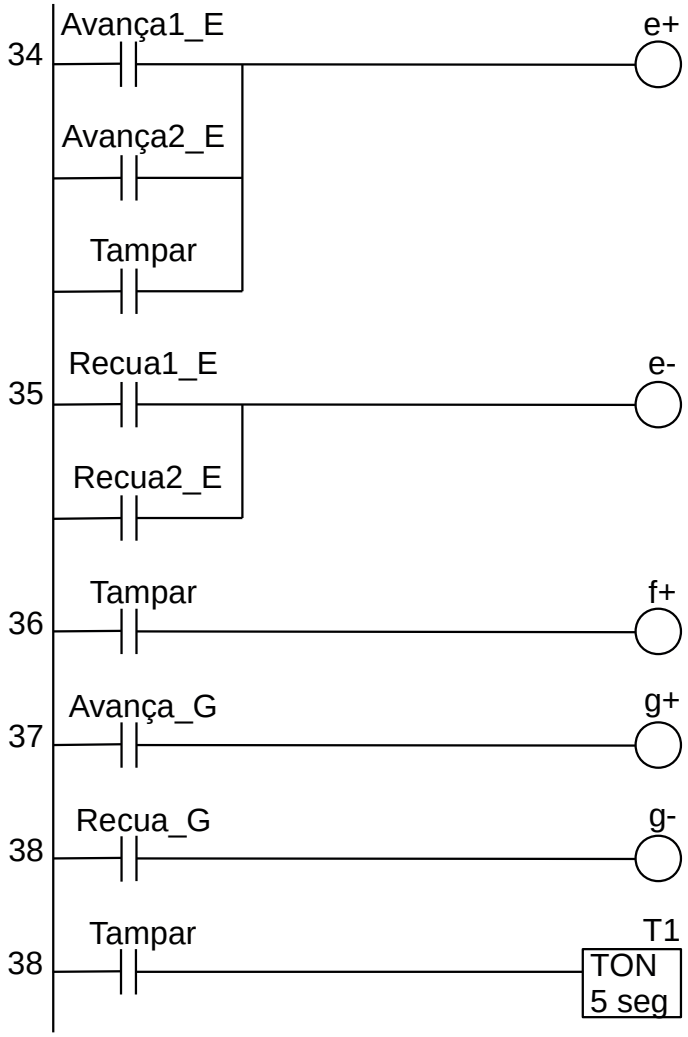

Figura 7.77 – Programa em linguagem Ladder para ativação das saídas da estação 3.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 7.10 e 7.11. As tabelas com as variáveis internas podem ser vistas nas tabelas 7.12.

Variável	Etiqueta	Descrição
M01	Espera1	Memória para estado.
M02	Avança_A	Memória para estado.
M03	Recua_A	Memória para estado.
M04	Avança_B	Memória para estado.
M05	Recua_B	Memória para estado.
M06	Espera2	Memória para estado.
M07	Avança_C	Memória para estado.
M08	Recua_C	Memória para estado.
M09	Espera2a	Memória para estado.
M10	Espera3	Memória para estado.

Tabela 7.12 – Descrição das variáveis internas (parte 1).

Variável	Etiqueta	Descrição
M11	Avança_G	Memória para estado.
M12	Avança1_E	Memória para estado.
M13	Recua1_E	Memória para estado.
M14	Recua_G	Memória para estado.
M15	Avança2_E	Memória para estado.
M16	Tampar	Memória para estado.
M17	Recua2_E	Memória para estado.
M18	Espera3a	Memória para estado.
M19	Log0	Resultado de frase lógica.
M20	Log1	Resultado de frase lógica.
M21	Log2	Resultado de frase lógica.
M22	Func	Memória de funcionamento do programa.
M20	Trava	Memória de trava de inicialização.
T01	T1	Temporizador para ligar (TON) 5 segundos.
C01	Cont	Contador regressivo. Valor = 6.

Tabela 7.12 – Descrição das variáveis internas (parte 2).

7.5.5 – Exemplo 5. Máquina de etiquetagem por carimbo

Esse exemplo é semelhante ao exemplo 2 do capítulo 6, mas com solução por diagrama de estado.

Automatizar a máquina de etiquetagem por carimbo mostrada na figura 7.78.

Descrição operacional da máquina ou processo

As caixas a serem etiquetadas chegam através de uma esteira de entrada do tipo acumuladora que possui o sensor “P4” identifica que existem 4 caixas prontas para serem etiquetadas.

O cilindro pneumático C1 empurra a caixa desde a esteira de entrada até a posição de etiquetagem sob o carimbo, conforme pode ser visto na região tracejada da figura 7.78. O cilindro C1 é acionado pelas eletroválvulas “C1+” para avanço do embolo do cilindro e “C1-” para recuo do embolo do cilindro. O sensor “C1i” identifica o embolo do cilindro C1 totalmente recuado e o sensor “C1f” identifica o embolo do cilindro C1 totalmente avançado.

O cilindro pneumático C2 está montado de modo perpendicular à mesa sobre a região quadrada tracejada indicada com a letra “X”, conforme pode ser visto na figura 7.78. O embolo do cilindro empurra o carimbo em direção à caixa. O cilindro C2 é acionado pelas eletroválvulas “C2+” para avanço do embolo do cilindro e “C2-” para recuo do embolo do cilindro. O sensor “C2i” identifica o embolo do cilindro C2 totalmente recuado e o sensor “C2f” identifica o embolo do cilindro C2 totalmente avançado.

A mesa de etiquetagem é lisa e permite que o cilindro C1 empurre a caixa sob a região de etiquetagem através de outra caixa. Quatro caixas são acumuladas na mesa.

O cilindro pneumático C3 empurra as 4 caixas desde a mesa de etiquetagem até a esteira de saída. O cilindro C3 é acionado pelas eletroválvulas “C3+” para avanço do embolo do cilindro e “C3-” para recuo do embolo do cilindro. O sensor “C3i” identifica o embolo do cilindro C3 totalmente recuado e o sensor “C3f” identifica o embolo do cilindro C3 totalmente avançado.

Existe um painel de controle com uma botoeira para iniciar o processo com etiqueta “PTD” (Partida) e uma botoeira para parar o processo com etiqueta “PRD” (Parada). Após a botoeira de parada ser ativada, o processo deve parar apenas quando as 4 caixas que estiverem sobre a mesa forem transferidas para a esteira de saída. Também existe uma lâmpada sinalizadora de alarme com etiqueta “ALM”.

Outras condições.

- Quando o controlador for energizado, todos os cilindros pneumáticos devem ser recuados. Se os sensores de início não forem todos ativados, o alarme deve ser ativado.
- O carimbo deve permanecer em contato com a caixa por 2 segundos.
- Quando o sensor “P4” permanecer ativo por 3 segundos indica que 4 caixas estão prontas para serem etiquetadas.
- O alarme deve ser ativado se o movimento de qualquer cilindro pneumático demorar mais 5 segundos.
- Na condição de alarme, todas as eletroválvulas são desligadas e a botoeira de Partida fica inoperante.

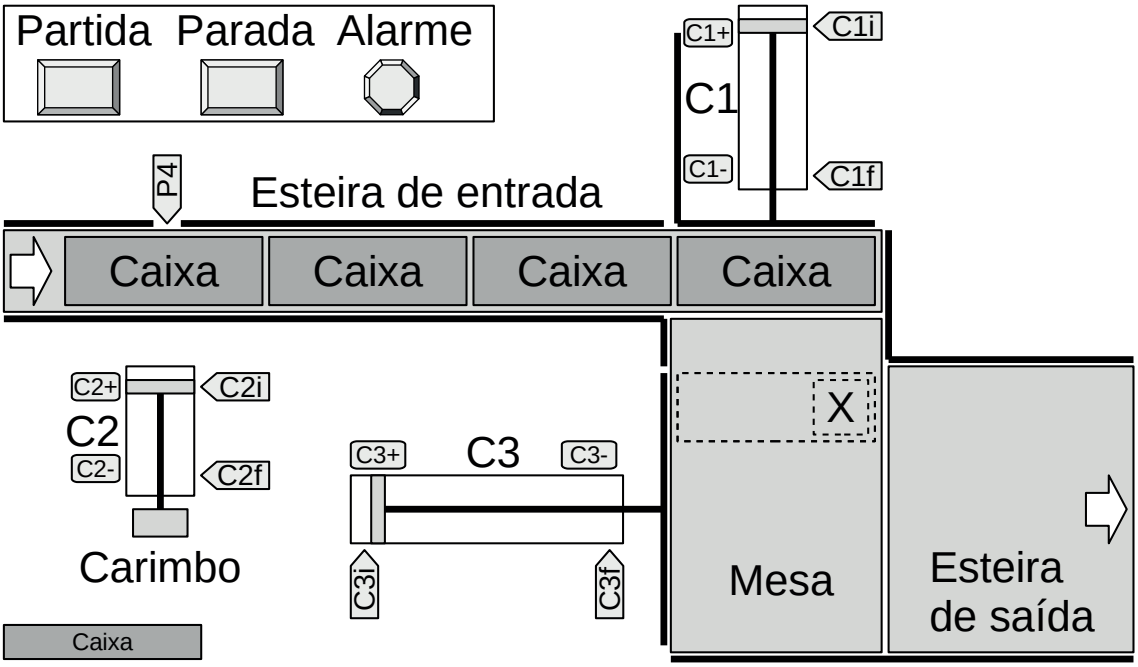

Figura 7.78 – Máquina de etiquetagem por carimbo.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entrada é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 7.13.

Parafuso	Etiqueta	Descrição
I01	PTD	Botoeira N.A. (Partida)
I02	PRD	Botoeira N.A. (Parada)
I03	P4	Sensor óptico de distância.
I04	C1i	Sensor magnético de cilindro pneumático.
I05	C1f	Sensor magnético de cilindro pneumático.
I06	C2i	Sensor magnético de cilindro pneumático.
I07	C2f	Sensor magnético de cilindro pneumático.
I08	C3i	Sensor magnético de cilindro pneumático.
I09	C3f	Sensor magnético de cilindro pneumático.

Tabela 7.13 – Descrição das entradas.

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 7.14.

Parafuso	**Etiqueta**	**Descrição**
Q01	C1+	Solenoide de eletroválvula pneumática.
Q02	C1-	Solenoide de eletroválvula pneumática.
Q03	C2+	Solenoide de eletroválvula pneumática.
Q04	C2-	Solenoide de eletroválvula pneumática.
Q05	C3+	Solenoide de eletroválvula pneumática.
Q06	C3-	Solenoide de eletroválvula pneumática.
Q07	ALM	Lâmpada sinalizadora. (Alarme)

Tabela 7.14 – Descrição das saídas.

Modelagem por diagrama de estados

O programa do controlador utiliza o método de diagrama de estados. O programa é dividido em três partes. A primeira parte prepara a máquina para as condições iniciais e verifica a possibilidade de ocorrência de erros para alarme. A segunda parte identifica se as 4 caixas estão prontas para serem etiquetadas e controla o movimento do cilindro C3. A segunda parte controla o movimento dos cilindros C1 e C2.

Esse programa exemplifica a técnica de interdependência cruzada, onde os dois programas são dependentes do outro programa.

O segundo programa possui um estado de espera chamado “Etiqueta2” que, quando ativado, vai habilitar uma transição no terceiro programa.

O terceiro programa possui um estado de espera chamado “Fim3” que, quando ativado, vai habilitar uma transição no segundo programa.

Os diagramas de estado para o controle funcional desta máquina pode ser visto na figura 5.79.

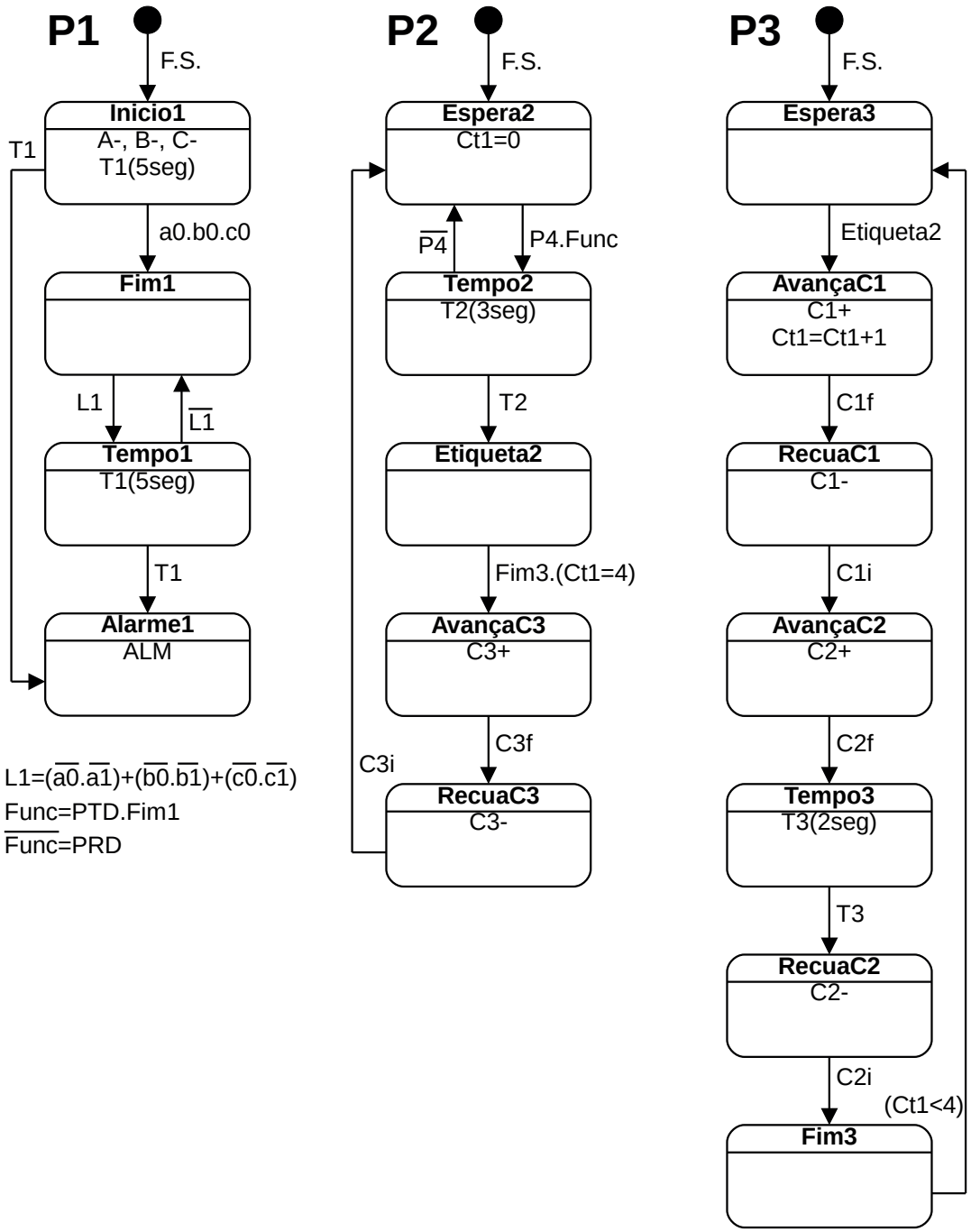

Figura 7.79 – Diagramas de estado para a máquina de etiquetagem por carimbo.

Descrição dos estados e transições

A ativação dos estados iniciais dos três diagramas (Início1, Espera2 e Espera3) é realizado pela técnica de primeiro ciclo (F.S.) usando o contato de bloqueio (trava).

Diagrama P1

Estado: Inicio1 Ação: A-, B-, C-, T1(5seg)

O controlador ativa as saídas digitais “A-” e “B-” e “C-” para recuo dos êmbolos dos cilindros pneumáticos e ativa o temporizador para ligar T1 de 5 segundos.

Transição: Inicio1 → Fim1 Lógica: a0.b0.c0

Todos os três cilindros pneumáticos estão com os êmbolos no início. O estado Inicio1 é desativado e o estado Fim1 é ativado.

Transição: Inicio1 → Alarme1 Lógica: T1

O temporizador T1 terminou a contagem antes que todos os êmbolos dos cilindros pneumáticos recuassem totalmente. O estado Inicio1 é desativado e o estado Alarme1 é ativado.

Estado: Fim1 Ação: nenhuma ação.

Com este estado ativo é possível ativar a variável interna de funcionamento (Func).

Transição: Fim1 → Tempo1 Lógica: $L1 = (\overline{a0}.\overline{a1})+(\overline{b0}.\overline{b1})+(\overline{c0}.\overline{c1})$

Um dos cilindros pneumáticos está em movimento. O estado Fim1 é desativado e o estado Tempo1 é ativado.

Estado: Tempo1 Ação: T1(5seg)

O controlador ativa o temporizador para ligar T1 de 5 segundos.

Transição: Tempo1 → Fim1 Lógica: $\overline{L1}$

Todos os cilindros pneumáticos estão parados ou no início ou no final de curso. O estado Tempo1 é desativado e o estado Fim1 é ativado.

Transição: Tempo1 → Alarme1 Lógica: T1

O temporizador T1 terminou a contagem antes que todos os êmbolos dos cilindros pneumáticos estejam parados no início ou no final de curso. O estado Tempo1 é desativado e o estado Alarme1 é ativado.

Estado: Alarme1 Ação: ALM

A saída de alarme é ativada.

Diagrama P2

Estado: Espera2 Ação: Ct1=0

O controlador zera o contador progressivo Ct1.

Transição: Espera2 → Tempo2 Lógica: $\overline{P4}$

Uma caixa passou em frente ao sensor P4. O estado Espera2 é desativado e o estado Tempo2 é ativado.

Estado: Tempo2 Ação: Temporizador T2 de 3 segundos.

O temporizador T2 é ativado para verificar se uma caixa ficou por 3 segundos na frente de P4.

Transição: Tempo2 → Espera2 Lógica: $\overline{P4}$

Uma caixa ficou menos de 3 segundos em frente ao sensor P4. O estado Tempo2 é desativado e o estado Espera2 é ativado.

Transição: Tempo2 → Etiqueta2 Lógica: T2

O temporizador terminou a contagem. O estado Tempo2 é desativado e o estado Etiqueta2 é ativado.

Estado: Etiqueta2 Ação: Nenhuma ação,

O estado é feito ativo para habilitar uma transição no terceiro programa.

Transição: Etiqueta2 → AvançaC3 Lógica: Fim3.(Ct1=4)

Quatro caixas já etiquetadas e estão prontas para serem empurradas. O estado Etiqueta2 é desativado e o estado AvançaC3 é ativado.

Estado: AvançaC3 Ação: C3+

O cilindro 3 é ativado para empurrar as 4 caixas etiquetadas.

Transição: AvançaC3 → RecuaC3 Lógica: C3f

O embolo do cilindro 3 chegou ao final de seu curso. O estado AvançaC3 é desativado e o estado RecuaC3 é ativado.

Estado: RecuaC3 Ação: C3-

O cilindro 3 é ativado para recuar o embolo para posição de início.

Transição: RecuaC3 → Espera2 Lógica: C3i

O embolo do cilindro 3 chegou ao início de seu curso. O estado RecuaC3 é desativado e o estado Espera2 é ativado.

Diagrama P3

Estado: Espera3 Ação: Nenhuma ação.

O controlador espera que um determinado estado do segundo programa fique ativo.

Transição: Espera3 → AvançaC1 Lógica: Etiqueta2

O estado Etiqueta2, do primeiro programa, está ativo. O estado Espera3 é desativado e o estado AvançaC1 é ativado.

Estado: AvançaC1 Ação: C1+

O cilindro 1 é ativado para empurrar 1 caixa para o local de etiquetagem.

Transição: AvançaC1 → RecuaC1 Lógica: C1f

O embolo do cilindro 1 chegou ao final de seu curso. O estado AvançaC1 é desativado e o estado RecuaC1 é ativado.

Estado: RecuaC1 Ação: C1-

O cilindro 1 é ativado para recuar o embolo para posição de início.

Transição: RecuaC1 → AvançaC2 Lógica: C1i

O embolo do cilindro 1 chegou ao início de seu curso. O estado RecuaC1 é desativado e o estado AvançaC2 é ativado.

Estado: AvançaC2 Ação: C2+

O cilindro 2 é ativado para empurrar o dispositivo de etiquetagem sobre a caixa.

Transição: AvançaC2 → Tempo3 Lógica: C2f

O embolo do cilindro 2 chegou ao final de seu curso. O estado AvançaC2 é desativado e o estado Tempo3 é ativado.

Estado: Tempo3 Ação: Temporizador T3 de 2 segundos.

O temporizador T3 é ativado para garantir o processo de etiquetagem.

Transição: Tempo3→ RecuaC2 Lógica: T3

O temporizador terminou a contagem. O estado Tempo3 é desativado e o estado RecuaC2 é ativado.

Estado: RecuaC2 Ação: C2-

O cilindro 2 é ativado para recuar o embolo para posição de início.

Transição: RecuaC2 → Fim3 Lógica: C2i

O embolo do cilindro 2 chegou ao início de seu curso. O estado RecuaC2 é desativado e o estado Fim3 é ativado.

Estado: Fim3 Ação: Nenhuma ação.

O estado é feito ativo para habilitar uma transição no segundo programa.

Transição: Fim3 → Espera3 Lógica: $\overline{Ct1}$

As quatro caixas etiquetadas foram empurradas por C3 no primeiro programa.

Codificação do diagrama de estados em linguagem Ladder

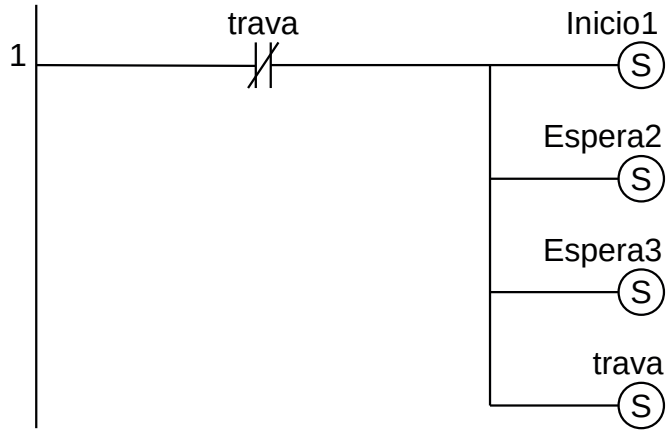

Figura 7.80 – Programa em linguagem Ladder para a inicialização.

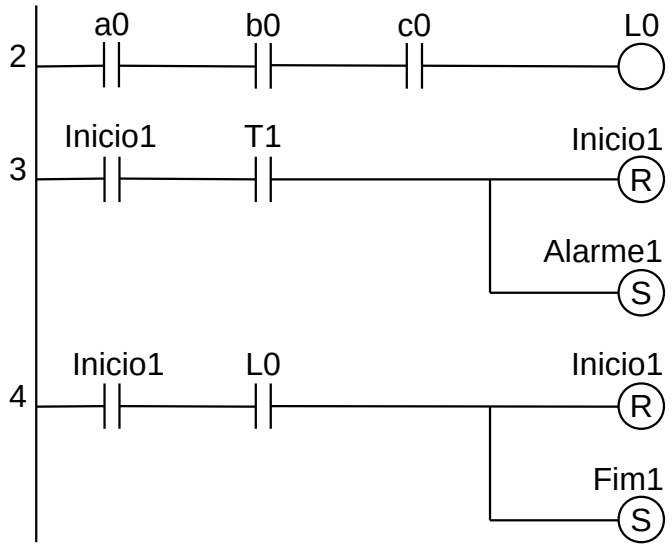

Figura 7.81 – Programa em linguagem Ladder para as transições saintes do estado Inicio1.

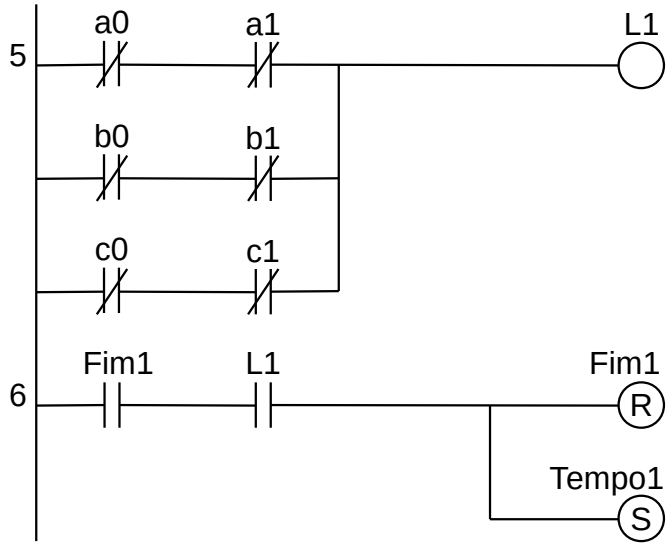

Figura 7.82 – Programa em linguagem Ladder para as transições saintes do estado Fim1.

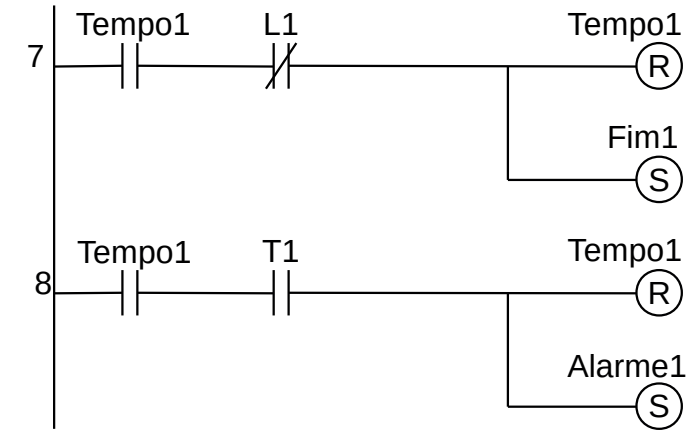

Figura 7.83 – Programa em linguagem Ladder para as transições saintes do estado Tempo1.

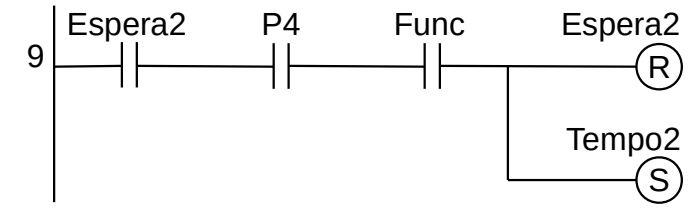

Figura 7.84 – Programa em linguagem Ladder para as transições saintes do estado Espera2.

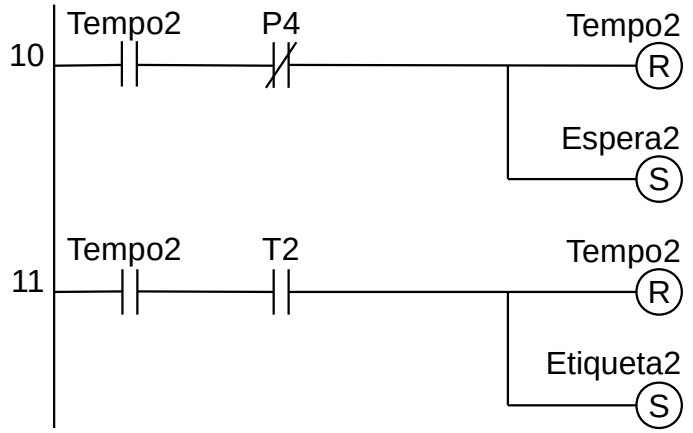

Figura 7.85 – Programa em linguagem Ladder para as transições saintes do estado Tempo2.

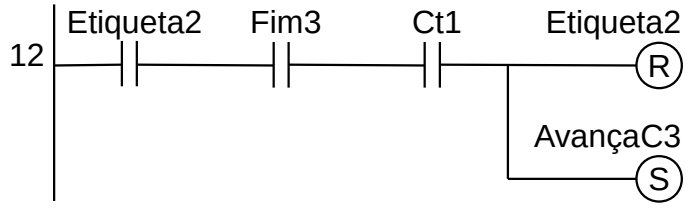

Figura 7.86 – Programa em linguagem Ladder para as transições saintes do estado Etiqueta2.

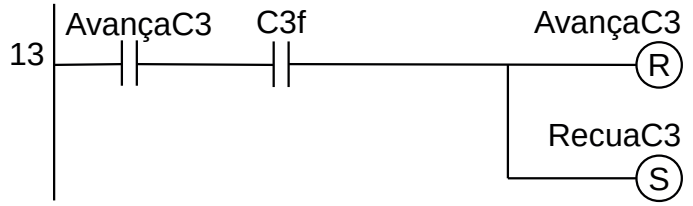

Figura 7.87 – Programa em linguagem Ladder para as transições saintes do estado AvançaC3.

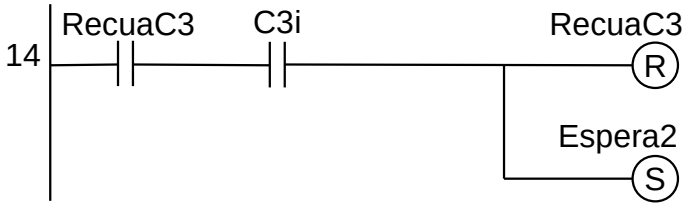

Figura 7.88 – Programa em linguagem Ladder para as transições saintes do estado RecuaC3.

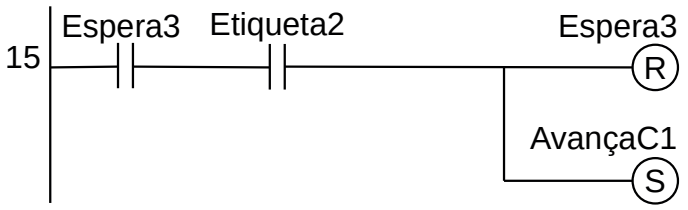

Figura 7.89 – Programa em linguagem Ladder para as transições saintes do estado Espera2.

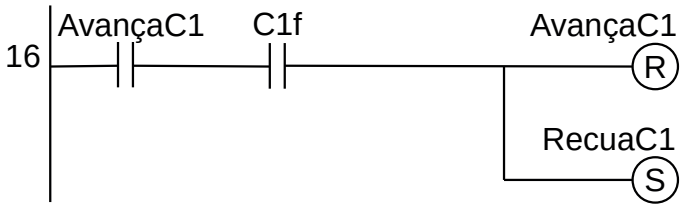

Figura 7.90 – Programa em linguagem Ladder para as transições saintes do estado AvançaC1.

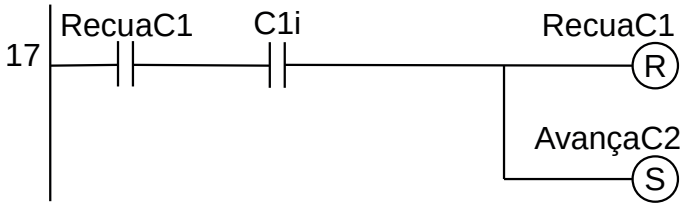

Figura 7.91 – Programa em linguagem Ladder para as transições saintes do estado RecuaC1.

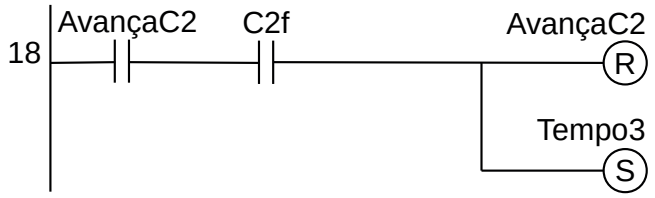

Figura 7.92 – Programa em linguagem Ladder para as transições saintes do estado AvançaC2.

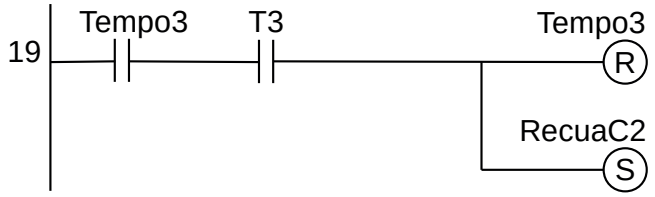

Figura 7.93 – Programa em linguagem Ladder para as transições saintes do estado Tempo3.

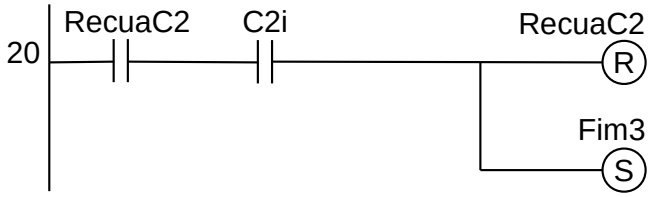

Figura 7.94 – Programa em linguagem Ladder para as transições saintes do estado RecuaC2.

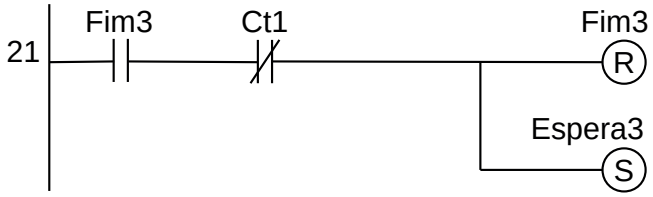

Figura 7.95 – Programa em linguagem Ladder para as transições saintes do estado Fim3.

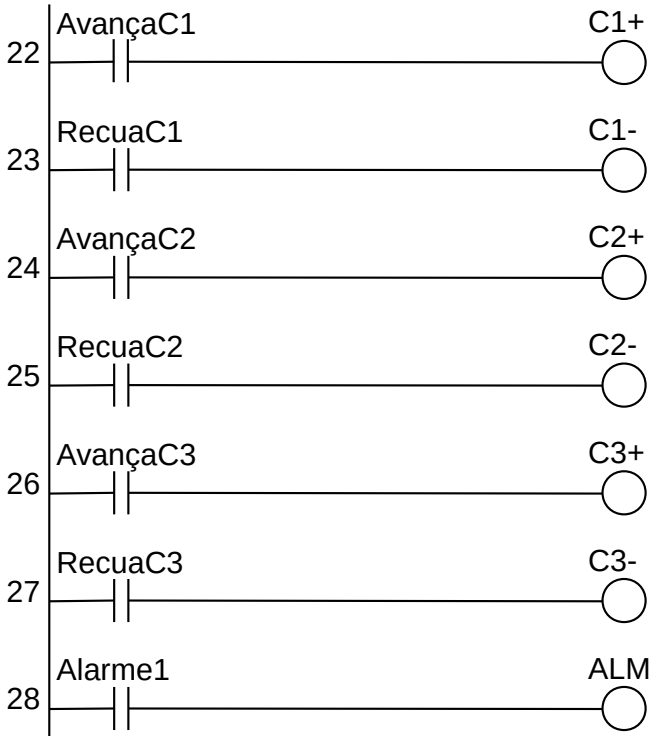

Figura 7.96 – Programa em linguagem Ladder para ativação das saídas digitais.

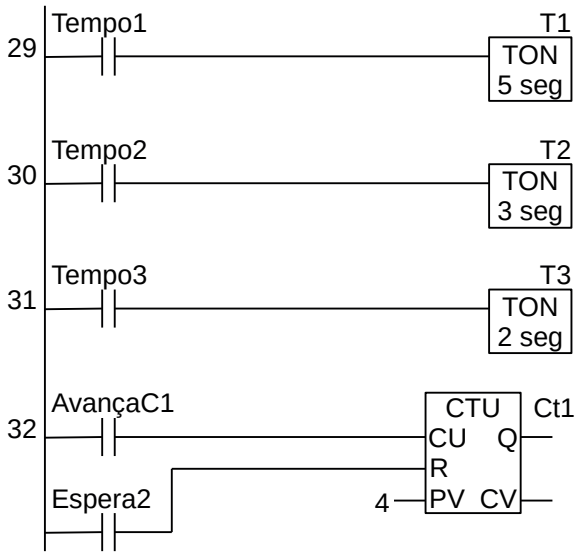

Figura 7.97 – Programa em linguagem Ladder para ativação dos temporizadores e contador.

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 7.13 e 7.14. A tabela com as variáveis internas pode ser vista na tabela 7.15.

Variável	Etiqueta	Descrição
M01	Inicio1	Memória para estado ativo.
M02	Fim1	Memória para estado ativo.
M03	Tempo1	Memória para estado ativo.
M04	Alarme1	Memória para estado ativo.
M05	Espera2	Memória para estado ativo.
M06	Tempo2	Memória para estado ativo.
M07	Etiqueta2	Memória para estado ativo.
M08	AvançaC3	Memória para estado ativo.
M09	RecuaC3	Memória para estado ativo.
M10	Espera3	Memória para estado ativo.
M11	AvançaC1	Memória para estado ativo.
M12	RecuaC1	Memória para estado ativo.
M13	AvançaC2	Memória para estado ativo.
M14	Tempo3	Memória para estado ativo.
M15	RecuaC2	Memória para estado ativo.
M16	Fim3	Memória para estado ativo.
M17	trava	Memória para trava de primeiro ciclo.
M18	L0	Variável interna.
M19	L1	Variável interna.
M20	Func	Variável interna.
T01	T1	Temporizador para ligar (TON) 5 segundos.
T02	T2	Temporizador para ligar (TON) 3 segundos.
T03	T3	Temporizador para ligar (TON) 2 segundos.
C01	Ct1	Contador progressivo

Tabela 7.15 – Descrição das variáveis internas.

7.5.6 – Exemplo 6. Máquina de furação automática

Esse exemplo mostra duas condições solicitadas: preparação da máquina e identificação de alarmes.

Automatizar a máquina de furação automática mostrada na figura 7.99.

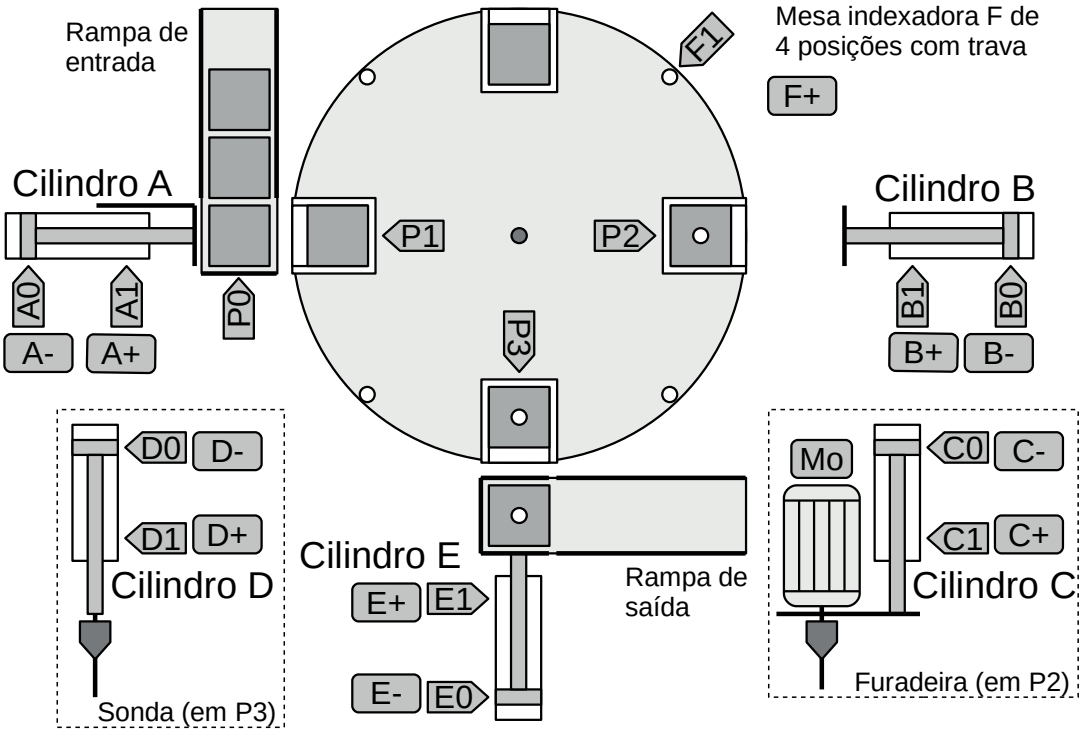

Figura 7.99 – Diagrama da máquina de furação automática.

A máquina é composta por 4 estações:

- **Estação 1** – Transferência. Cilindro pneumático A. Tem a função de transferir uma peça da rampa de entrada para a mesa indexadora.
- **Estação 2** – Furação. Cilindro pneumático B, cilindro pneumático C e furadeira. Tem a função de prender a peça e realizar o furo.
- **Estação 3** – Testagem. Cilindro pneumático D e cilindro pneumático E. Tem a função de testar a realização do furo e extrair a peça.
- **Estação 4** – Girar. Mesa indexadora F. Tem a função de transferir uma peça de uma estação para outra.

Descrição da máquina

Uma máquina para furação é composta pelos seguintes componentes.

a) Uma rampa de entrada por onde as peças a serem furadas escorregam por gravidade até um anteparo.

b) Um sensor de posição tipo indutivo com etiqueta “P0” que identifica uma peça na rampa pronta para ser colocada na ranhura da mesa indexadora pelo cilindro A.

c) Um cilindro pneumático de dupla ação (A) que empurra a peça desde a rampa de entrada para uma ranhura na mesa indexadora. O cilindro A possui os solenoides com etiquetas “A+” para avançar e “A-” para recuar e sensores magnéticos de fim-de-curso “A1” e início-de-curso “A0”.

d) Um sensor de posição tipo indutivo com etiqueta “P1” que identifica uma peça colocada na ranhura da mesa indexadora pelo cilindro A.

e) Uma mesa indexadora pneumática (F) com 4 ranhuras para acomodar as peças em processo de furação. A mesa indexadora F possui o solenoide com etiqueta “F+” para fazer o giro da mesa e um sensor tipo fim-de-curso com etiqueta “F1” de detecta quando a mesa girou 90 graus. Observe que “F1” é falso quando a mesa indexadora está em processo de giro e é verdadeiro quando a mesa girou 90 graus.

f) Um cilindro pneumático de dupla ação (B) que prende a peça na ranhura na mesa indexadora para furação. O cilindro B possui os solenoides com etiquetas “B+” para avançar e “B-” para recuar e sensores magnéticos de fim-de-curso “B1” e início-de-curso “B0”.

g) Um sensor de posição tipo indutivo com etiqueta “P2” que identifica uma peça colocada na ranhura da mesa indexadora na posição da furadeira.

h) Um cilindro pneumático de dupla ação (C) que movimenta o motor de furação em direção à peça presa na mesa indexadora. O cilindro C possui os solenoides com etiquetas “C+” para avançar e “C-” para recuar e sensores magnéticos de fim-de-curso “C1” e início-de-curso “C0”.

i) Uma furadeira com motor de indução acionado por um contator com etiqueta “Mo”.

j) Um cilindro pneumático de dupla ação (D) que movimenta uma sonda de profundidade em direção à peça presa na mesa indexadora. O cilindro D possui os solenoides com etiquetas “D+” para avançar e “D-” para recuar e sensores magnéticos de fim-de-curso “D1” e início-de-curso “D0”.

k) Um sensor de posição tipo indutivo com etiqueta “P3” que identifica uma peça colocada na ranhura da mesa indexadora na posição da sonda de profundidade.

l) Um cilindro pneumático de dupla ação (E) que puxa a peça da mesa indexadora para a rampa de saída. O cilindro E possui os solenoides com etiquetas “E+” para avançar e “E-” para recuar e sensores magnéticos de fim-de-curso “E1” e início-de-curso “E0”.

m) Uma lâmpada sinalizadora de alarme com etiqueta “Alm1” para falha de inicialização. Uma lâmpada sinalizadora de alarme com etiqueta “Alm2” para falha de furação. Uma lâmpada sinalizadora de alarme com etiqueta “Alm3” para falha de teste de furação.

Considerações

a) A mesa indexadora gira no sentido horário.

b) Deve ser verificado a existência de peças sobre as 4 posições da mesa indexadora no instante da ativação do programa. Se verdadeiro, o alarme “Alm1” deve ser acionado e a máquina não pode operar.

c) Ao iniciar a ativação do programa, todos os cilindro pneumáticos devem ser recuados, a eletroválvula da mesa indexadora deve ser desligada e o contator do motor da furadeira deve ser desenergizado.

d) Uma botoeira com etiqueta “Partida” é utilizada para iniciar e reiniciar o funcionamento da máquina. Uma botoeira com etiqueta “Parada” é utilizada para parar a máquina. O evento de parada deve esvaziar todas as ranhuras da mesa indexadora.

e) O motor da furadeira deve estar em funcionamento durante o período de descida e subida.

f) O tempo máximo para descida da broca sobre a peça a ser furada é de 5 segundos. Se o tempo for excedido, o alarme “Alm2” deve ser acionado e a máquina não pode operar.

g) O tempo máximo para descida da sonda sobre a peça furada é de 3 segundos. Se o tempo for excedido, o alarme “Alm3” deve ser acionado e a máquina não pode operar.

h) O cilindro pneumático E deve estar avançado para poder extrair a peça furada da mesa indexadora. O gancho extrator de peças deve ficar posicionado sobre a rampa de saída por 3 segundos para garantir que a peça saia totalmente do gancho extrator.

Funcionamento da máquina

Estação 1 – *Transferência*

a) Uma peça escorrega pela rampa de entrada e ativa o sensor “P0”.
b) Se a baia da mesa indexadora estiver vazia, o sensor “P1” é falso.
c) O cilindro A empurra a peça até a baia da mesa indexadora.
d) O cilindro A recua.

Estação 2 – *Furação*

a) Uma peça chega na estação, o sensor “P2” é verdadeiro.
b) O cilindro B avança e prende a peça na baia da mesa indexadora.
c) O motor da furadeira é ligado.
d) O cilindro C avança e executa a furação na peça.
e) O cilindro C recua.
f) O motor da furadeira é desligado.
g) O cilindro B recua.

Estação 3 – *Testagem*

a) O cilindro E avança.
b) Uma peça chega na estação, o sensor “P3” é verdadeiro.
c) O cilindro D avança.
d) O cilindro D recua.
e) O cilindro E recua.
f) Um temporizador de 3 segundos é ativado.

Estação 4 – *Girar*

a) As três estações estão paradas e existe uma peça na mesa indexadora.
b) A mesa indexadora realiza um giro de 90 graus.

Solução da automação da máquina ou processo

Para solucionar este problema é necessário montar as tabelas de entrada e saída, identificando os pinos do controlador nos quais serão ligados os dispositivos de sensores e atuadores encontrados na máquina ou processo.

A tabela de entradas é montada com a descrição dos sensores ou outros dispositivos de coleta de informações da máquina ou processo. A tabela das entradas pode ser vista na tabela 7.16.

Parafuso	Etiqueta	Descrição
I01	Partida	Botoeira NA.
I02	Parada	Botoeira NA.
I03	A0	Sensor magnético tipo “reed”.
I04	A1	Sensor magnético tipo “reed”.
I05	B0	Sensor magnético tipo “reed”.
I06	B1	Sensor magnético tipo “reed”.
I07	C0	Sensor magnético tipo “reed”.
I08	C1	Sensor magnético tipo “reed”.

Tabela 7.16 – Descrição das entradas (parte 1).

Parafuso	Etiqueta	Descrição
I09	D0	Sensor magnético tipo “reed”.
I10	D1	Sensor magnético tipo “reed”.
I11	E0	Sensor magnético tipo “reed”.
I12	E1	Sensor magnético tipo “reed”.
I13	F1	Sensor magnético tipo “reed”.
I14	P0	Sensor de posição tipo indutivo.
I15	P1	Sensor de posição tipo indutivo.
I16	P2	Sensor de posição tipo indutivo.
I17	P3	Sensor de posição tipo indutivo.

Tabela 7.16 – Descrição das entradas (parte 2).

A tabela de saídas é montada com a descrição dos atuadores ou outros dispositivos de envio de informações para a máquina ou processo. A tabela das saídas pode ser vista na tabela 7.17.

Parafuso	Etiqueta	Descrição
Q01	A+	Solenoide de eletroválvula pneumática.
Q02	A-	Solenoide de eletroválvula pneumática.
Q03	B+	Solenoide de eletroválvula pneumática.
Q04	B-	Solenoide de eletroválvula pneumática.
Q05	C+	Solenoide de eletroválvula pneumática.
Q06	C-	Solenoide de eletroválvula pneumática.
Q07	D+	Solenoide de eletroválvula de líquido.
Q08	D-	Solenoide de eletroválvula pneumática.
Q09	E+	Solenoide de eletroválvula pneumática.
Q10	E-	Solenoide de eletroválvula pneumática.
Q11	F+	Solenoide de eletroválvula pneumática.
Q12	Alm1	Lâmpada sinalizadora.
Q13	Alm2	Lâmpada sinalizadora.
Q14	Alm3	Lâmpada sinalizadora.
Q15	Mo	Contator para motor monofásico.

Tabela 7.17 – Descrição das saídas.

Parte 1 – Inicialização

Esta parte atende às condições operacionais exigidas antes da máquina entrar em funcionamento normal para o processo de furação e teste das peças.

Essas condições de início são:

- Nenhuma peça deve estar nas baias da mesa indexadora.
- Todos os cilindros pneumáticos devem estar recuados.

Um diagrama de estados é elaborado para implementar essas exigências e pode ser visto na figura 7.100.

Uma frase lógica produz um “pulso único” após o início de execução do programa do controlador. Esse pulso ativa o primeiro bloco do diagrama de estado.

SE (“trava1” é falso) ENTÃO (fazer “Início” verdadeiro e memorizar e fazer “trava1” verdadeiro e memorizar)

Detalhamento dos estados e transições

Estado: “Início” Ação: Nenhuma ação.

Este é um estado de espera para verificar a posição da mesa indexadora. Se a mesa não estiver posicionada (F1 falso) é direcionado ao término do posicionamento.

Transição: “Início” → “GiraF1” Lógica: “$\overline{F1}$”

Se a mesa não estiver posicionada corretamente, F1 é falso, e deve ser direcionado para o término de giro da mesa indexadora. O estado “Início” é desativado e o estado “GiraF1” é ativado.

Transição: “Início” → “Teste1” Lógica: “F1”

A mesa indexadora está na posição e é executado o teste de verificação de peças em 3 baias da mesa indexadora. O estado “Início” é desativado e o estado “Teste1” é ativado.

Estado: “GiraF1” Ação: “F+”

A eletroválvula pneumática “F+” é ativada para girar a mesa indexadora.

Transição: “GiraF1” → “Teste1” Lógica: “F1”

A mesa indexadora está posicionada corretamente. O estado “GiraF1” é desativado e o estado “Teste1” é ativado.

Estado: “Teste1” Ação: Nenhuma ação.

Este é um estado de espera para verificar a condição de existência ou não de peças em 3 baias da mesa indexadora.

Transição: “Teste1” → “RecuaD1” Lógica: “$\overline{P1}.\overline{P2}.\overline{P3}$”

Nenhuma das 3 baias com sensores possui uma peça. O estado “Teste1” é desativado e o estado “RecuaD1” é ativado.

Transição: “Teste1” → “Alarme1” Lógica: “P1+P2+P3”

Pelo menos uma das 3 baias com sensores possui uma peça. O estado “Teste1” é desativado e o estado “Alarme1” é ativado.

Estado: “Alarme1” Ação: “Alm1”

A condição de alarme de inicialização é ativada. O programa não evolui para outro estado.

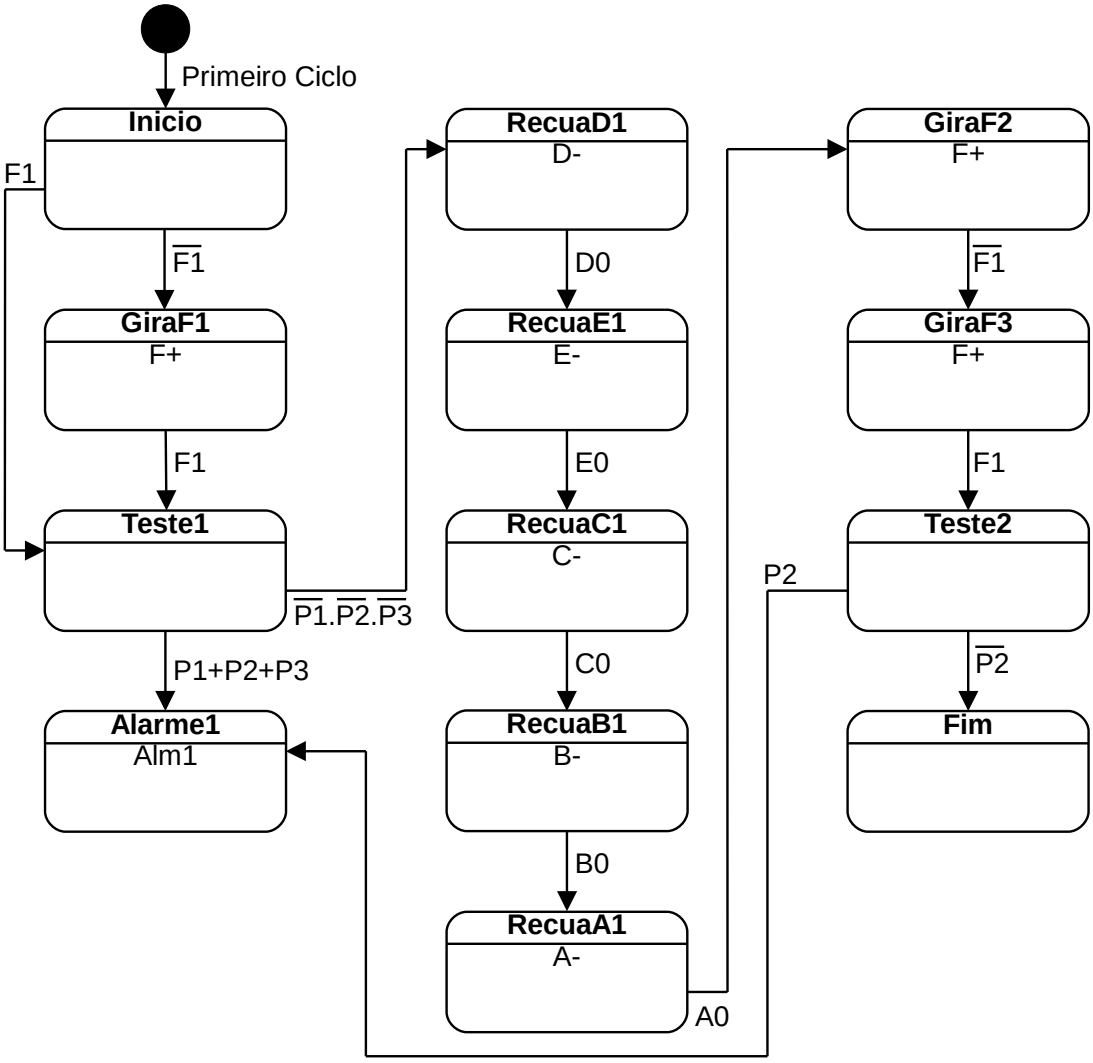

Figura 7.100 – Diagrama de estados para a inicialização da máquina de furação.

Estado: “RecuaD1” Ação: “D-”

A eletroválvula pneumática “D-” é ativada para recuar o embolo do cilindro D.

Transição: “RecuaD1” → “RecuaE1” Lógica: “D0”

O embolo do cilindro D foi totalmente recuado. O estado “Recuad1” é desativado e o estado “RecuaE1” é ativado.

Estado: “RecuaE1” Ação: “E-”

A eletroválvula pneumática “E-” é ativada para recuar o embolo do cilindro E.

Transição: “RecuaE1” → “RecuaC1” Lógica: “E0”

O embolo do cilindro E foi totalmente recuado. O estado “RecuaE1” é desativado e o estado “RecuaC1” é ativado.

Estado: “RecuaC1” Ação: “C-”

A eletroválvula pneumática “C-” é ativada para recuar o embolo do cilindro C.

Transição: “RecuaC1” → “RecuaB1” Lógica: “C0”

O embolo do cilindro C foi totalmente recuado. O estado “RecuaC1” é desativado e o estado “RecuaB1” é ativado.

Estado: “RecuaB1” Ação: “B-”

A eletroválvula pneumática “B-” é ativada para recuar o embolo do cilindro B.

Transição: “RecuaB1” → “RecuaA1” Lógica: “B0”

O embolo do cilindro B foi totalmente recuado. O estado “RecuaB1” é desativado e o estado “RecuaA1” é ativado.

Estado: “RecuaA1” Ação: “A-”

A eletroválvula pneumática “A-” é ativada para recuar o embolo do cilindro A.

Transição: “RecuaA1” → “GiraF2” Lógica: “A0”

O embolo do cilindro A foi totalmente recuado. O estado “RecuaA1” é desativado e o estado “GiraF2” é ativado.

Estado: “GiraF2” Ação: “F+”

A eletroválvula pneumática “F+” é ativada para girar a mesa indexadora.

Transição: “GiraF2” → “GiraF3” Lógica: “$\overline{F1}$”

A mesa indexadora começou a girar e saiu da posição de trava. O estado “GiraF2” é desativado e o estado “GiraF3” é ativado.

Estado: “GiraF3” Ação: “F+”

A eletroválvula pneumática “F+” é ativada para girar a mesa indexadora.

Transição: “GiraF3” → “Teste2” Lógica: “F1”

A mesa indexadora terminou o giro de 90 graus e atingiu a posição de trava. O estado “GiraF3” é desativado e o estado “Teste2” é ativado.

Estado: “Teste2” Ação: Nenhuma ação.

Este é um estado de espera para verificar a condição de existência ou não uma peça em uma baia da mesa indexadora.

Transição: “Teste2” → “Alarme1” Lógica: “P2”

Existe uma peça na baia do sensor P2. O estado “Teste2” é desativado e o estado “Alarme1” é ativado.

Transição: “Teste2” → “Fim” Lógica: “$\overline{P2}$”

Não existe uma peça na baia do sensor P2. O estado “Teste2” é desativado e o estado “Fim” é ativado.

Estado: “Fim” Ação: Nenhuma ação.

Este é um estado de espera para ativar os programas de funcionamento normal da máquina de furação automática. O programa não evolui para outro estado.

Parte 2 – Funcionamento normal

É criado a variável “Func” que indica que o programa do controlador pode ser executado.

Esta variável é controlada pelas botoeiras “Partida” e “Parada”. Também é criada a variável “trava2” para inibir o reinício do primeiro bloco do diagrama de estados através da botoeira de inicialização.

É utilizado frases lógicas para a modelagem dessa etapa.

A primeira frase produz um “pulso único” a partir do estado “Fim” do programa da parte de inicialização. Esse pulso ativa os primeiros blocos dos diagramas de estados de cada estação.

> SE (“Fim” é verdadeiro e “trava2” é falso) ENTÃO (fazer “InícioA” verdadeiro e memorizar e fazer “InícioBC” verdadeiro e memorizar e fazer “AvançaE” verdadeiro e memorizar e fazer “EsperaF” verdadeiro e memorizar e fazer “trava2” verdadeiro e memorizar)

A segunda frase permite a ativação da variável “Func”, habilitando o funcionamento do programa.

> SE (“Partida” é verdadeiro) ENTÃO (fazer a variável “Func” verdadeiro e memorizar).

A terceira frase desabilita a variável de funcionamento do programa.

> SE (“Parada” é verdadeiro) ENTÃO (fazer a variável “Func” falso).

Cada um dos 4 programas possui estados de espera para início das atividades.

O diagrama de estados para solucionar este problema pode ser visto na figura 7.101.

- O programa 1, que faz a colocação de uma peça na baia da mesa indexadora, fica aguardando a existência de uma peça na rampa de entrada (P0) e a mesa indexadora posicionada (F1) e a variável de funcionamento (Func) ativa.
- O programa 2, que executa a furação na peça, fica aguardando a existência de uma peça sob a furadeira (P2) e a mesa indexadora posicionada (F1).
- O programa 3, que executa o teste de furação, fica aguardando a existência de uma peça sob o testador (P3) e a mesa indexadora posicionada (F1).

- O programa 4, que executa o giro da mesa indexadora, fica aguardando que os outros 3 programas estejam parados em algum estado de espera e que o valor de um contador seja maior que zero.

É criado a variável “Cont” para um contador regressivo. Esse contador é iniciado com o número 3 quando uma peça é transferida da rampa de entrada para a baia da mesa indexadora. Esse contador é decrementado de uma unidade quando a mesa indexadora executa um giro de 90 graus. O objetivo é proporcionar o esvaziamento das baias da mesa indexadora quando da falta de peças.

Programa 1 – transferência de uma peça da rampa de entrada para a baia da mesa indexadora

Estado: “InícioA” Ação: Nenhuma ação.

O controlador espera por uma peça na rampa de entrada e a condição para colocação da peça na baia da mesa indexadora.

Transição: “InícioA” → “AvançaA” Lógica: “F1.P0.Func”

Existe uma peça na rampa de entrada (P0) e condição verdadeira para colocação da peça na baia da mesa indexadora (F1.Func). O estado “InícioA” é desativado e o estado “AvançaA” é ativado.

Estado: “AvançaA” Ação: “A+” e “Cont=3”

A eletroválvula “A+” é ativada para permitir o embolo do cilindro A empurrar a peça da rampa de entrada para a baia da mesa indexadora. O contador “Cont” é iniciado com o valor 3.

Transição: “AvançaA” → “RecuaA” Lógica: “A1”

O embolo do cilindro A chegou a final do curso e o sensor “A1” torna-se verdadeiro. O estado “AvançaA” é desativado e o estado “RecuaA” é ativado.

Estado: “RecuaA” Ação: “A-”

A eletroválvula “A-” é ativada para permitir o recuo embolo do cilindro A para posição de início.

Transição: “RecuaA” → “FimA” Lógica: “A0”

O embolo do cilindro A chegou a início do curso e o sensor “A0” torna-se verdadeiro. O estado “RecuaA” é desativado e o estado “FimA” é ativado.

Estado: “FimA” Ação: Nenhuma ação.

O controlador fica aguardando a baia da mesa indexadora de frente do cilindro A ficar vazia.

Transição: “FimA” → “InícioA” Lógica: “$\overline{P1}$”

A baia da mesa indexadora de frente do cilindro A fica vazia e o sensor “P1” torna-se falso. O estado “FimA” é desativado e o estado “InícioA” é ativado.

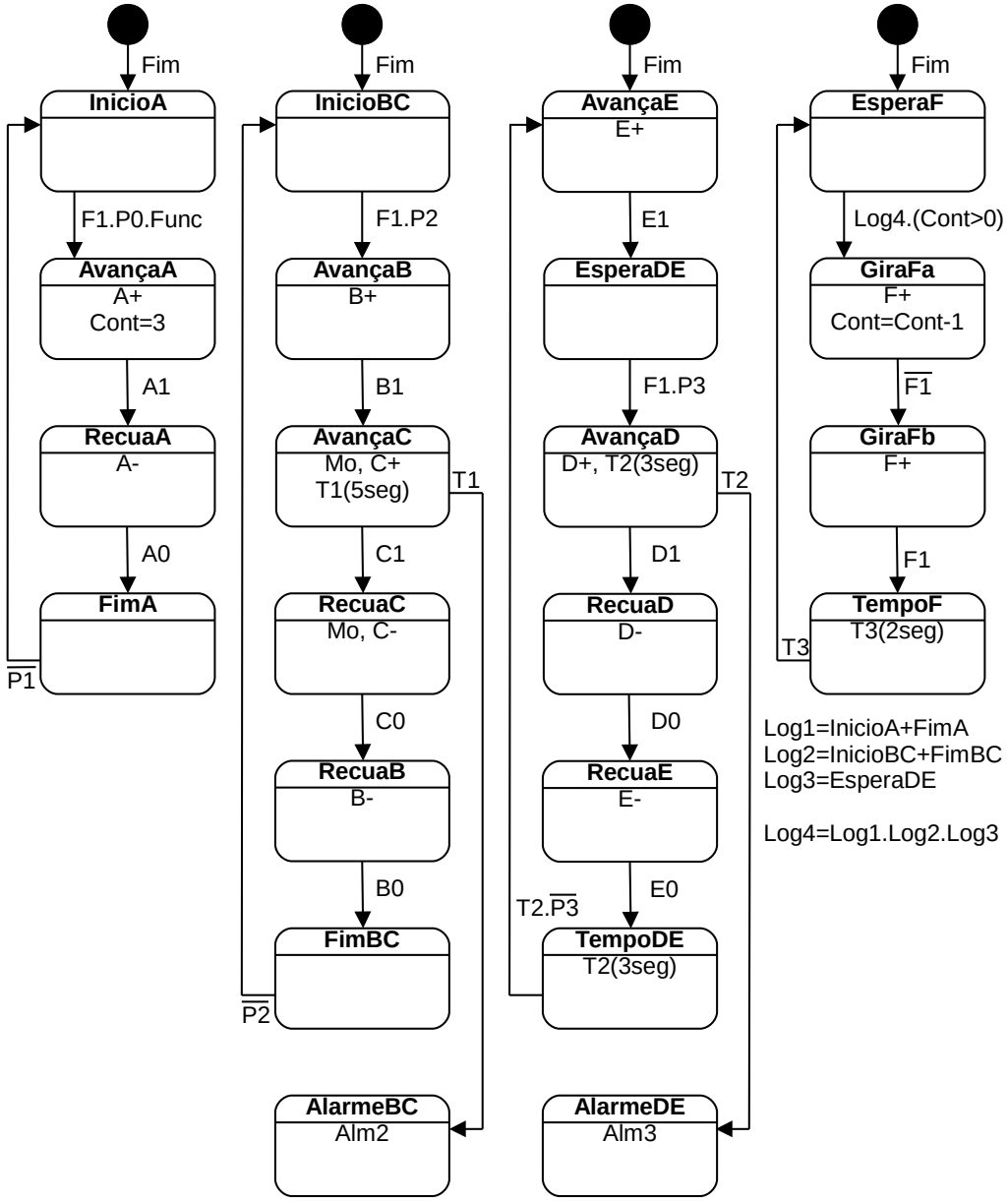

Figura 7.101 – Diagrama de estados para o controle da máquina de furação.

Programa 2 – furação da peça

Estado: “InícioBC” Ação: Nenhuma ação.

O controlador espera por uma peça na baia da mesa indexadora de frente ao cilindro B.

Transição: “InícioBC” → “AvançaB” Lógica: “F1.P2”

Existe uma peça na baia da mesa indexado de frente ao cilindro B (P2) e condição verdadeira para furação da peça (F1). O estado “InícioBC” é desativado e o estado “AvançaB” é ativado.

Estado: “AvançaB” Ação: “B+”

A eletroválvula “B+” é ativada para permitir o embolo do cilindro B prender a peça na baia da mesa indexadora.

Transição: “AvançaB” → “AvançaC” Lógica: “B1”

O embolo do cilindro B chegou a final do curso e o sensor “B1” torna-se verdadeiro. O estado “AvançaB” é desativado e o estado “AvançaC” é ativado.

Estado: “AvançaC” Ação: “C+” e “Mo” e “temporizador T1 de 5 seg.”

A eletroválvula “C+” é ativada para permitir o embolo do cilindro C mover a furadeira em direção à peça. O motor da furadeira é ligado. O temporizador de 5 segundos é ligado.

Transição: “AvançaC” → “RecuaC” Lógica: “C1”

O embolo do cilindro C chegou a final do curso e o sensor “C1” torna-se verdadeiro. O estado “AvançaC” é desativado e o estado “RecuaC” é ativado.

Transição: “AvançaC” → “AlarmeBC” Lógica: “T1”

O temporizador T1 terminou a contagem antes da broca da furadeira terminar o furo. Uma das possíveis causas é a broca estar “cega” e não executar o processo de furação. O estado “AvançaC” é desativado e o estado “AlarmeBC” é ativado.

Estado: “RecuaC” Ação: “C-” e “Mo”

A eletroválvula “C-” é ativada para permitir o recuo embolo do cilindro C para posição de início. O motor da furadeira permanece ligado.

Transição: “RecuaC” → “RecuaB” Lógica: “C0”

O embolo do cilindro C chegou a início do curso e o sensor “C0” torna-se verdadeiro. O estado “RecuaC” é desativado e o estado “RecuaB” é ativado.

Estado: “RecuaB” Ação: “B-”

A eletroválvula “B-” é ativada para permitir o recuo embolo do cilindro B para posição de início.

Transição: “RecuaB” → “FimBC” Lógica: “B0”

O embolo do cilindro B chegou a início do curso e o sensor “B0” torna-se verdadeiro. O estado “RecuaB” é desativado e o estado “FimBC” é ativado.

Estado: “FimBC” Ação: Nenhuma ação.

O controlador fica aguardando a baia da mesa indexadora de frente do cilindro B ficar vazia.

Transição: “FimBC” → “InícioBC” Lógica: “$\overline{P2}$”

A baia da mesa indexadora de frente do cilindro B fica vazia e o sensor “P2” torna-se falso. O estado “FimBC” é desativado e o estado “InícioBC” é ativado.

Estado: “AlarmeBC” Ação: “Alm2”

A condição de alarme de falha de furação é ativada.

Programa 3 – teste de furação

Estado: “AvançaE” Ação: “E+”

A eletroválvula “E+” é ativada para permitir o embolo do cilindro E avançar sobre a baia da mesa indexadora para permitir puxar a peça furada.

Transição: “AvançaE” → “EsperaDE” Lógica: “E1”

O embolo do cilindro E chegou a final do curso e o sensor “E1” torna-se verdadeiro. O estado “AvançaE” é desativado e o estado “EsperaDE” é ativado.

Estado: “EsperaDE” Ação: Nenhuma ação.

O controlador espera por uma peça na baia da mesa indexadora de frente ao cilindro E.

Transição: “EsperaDE” → “AvançaD” Lógica: “F1.P3”

Existe uma peça na baia da mesa indexado de frente ao cilindro E (P3) e condição verdadeira para furação da peça (F1). O estado “EsperaDE” é desativado e o estado “AvançaD” é ativado.

Estado: “AvançaD” Ação: “D+” e “temporizador T2 de 3 seg.”

A eletroválvula “D+” é ativada para permitir o embolo do cilindro D mover a sonda de teste em direção à peça. O temporizador de 3 segundos é ligado.

Transição: “AvançaD” → “RecuaD” Lógica: “D1”

O embolo do cilindro D chegou a final do curso e o sensor “D1” torna-se verdadeiro. O estado “AvançaD” é desativado e o estado “RecuaD” é ativado.

Transição: “AvançaD” → “AlarmeDE” Lógica: “T2”

O temporizador T2 terminou a contagem antes da sonda chegar ao final do furo. Uma das possíveis causas é a broca ter quebrado. O estado “AvançaD” é desativado e o estado “AlarmeDE” é ativado.

Estado: “RecuaD” Ação: “D-”

A eletroválvula “D-” é ativada para permitir o recuo embolo do cilindro D para posição de início.

Transição: “RecuaD” → “RecuaE” Lógica: “D0”

O embolo do cilindro D chegou a início do curso e o sensor “D0” torna-se verdadeiro. O estado “RecuaD” é desativado e o estado “RecuaE” é ativado.

Estado: “RecuaE” Ação: “E-”

A eletroválvula “E-” é ativada para permitir o recuo embolo do cilindro E para posição de início.

Transição: “RecuaE” → “TempoDE” Lógica: “E0”

O embolo do cilindro E chegou a início do curso e o sensor “E0” torna-se verdadeiro. O estado “RecuaE” é desativado e o estado “TempoDE” é ativado.

Estado: “TempoDE” Ação: “temporizador T2 de 3 seg.”

O controlador espera o tempo de 3 segundos para a peça furada escorregar pela rampa de saída.

Transição: “TempoDE” → “AvançaE” Lógica: “ $T2.\overline{P3}$ ”

O temporizador T2 terminou a contagem e a peça já saiu da baia da mesa indexadora. O estado “TempoDE” é desativado e o estado “AvançaE” é ativado.

Estado: “AlarmeDE” Ação: “Alm3”

A condição de alarme de falha de teste de furação é ativada.

Programa 4 – giro da mesa indexadora

Estado: “EsperaF” Ação: Nenhuma ação.

O controlador espera pela condição de giro, que é quando os programas 1, 2 e 3 estão em estados de espera e o contador de peças está em maior que 3.

Transição: “EsperaF” → “GiraF1” Lógica: “Log4”

A condição para giro da mesa indexadora é obtida quando os programas 1, 2 e 3 estão em estados de espera e o contador de peças está em maior que 3. O estado “EsperaF” é desativado e o estado “GiraF1” é ativado.

A variável “Log4” é obtida por uma frase lógica. Outras variáveis são usadas para uniformizar o entendimento da lógica.

SE (“InícioA” é verdadeiro ou “FimA” é verdadeiro) ENTÃO (fazer a variável “Log1” verdadeiro).

SE (“InícioBC” é verdadeiro ou “FimBC” é verdadeiro) ENTÃO (fazer a variável “Log2” verdadeiro).

SE (“EsperaDE” é verdadeiro) ENTÃO (fazer a variável “Log3” verdadeiro).

SE (“Log1” é verdadeiro e “Log2” é verdadeiro e “Log3” é verdadeiro e Cont>0) ENTÃO (fazer a variável “Log4” verdadeiro).

Estado: “GiraF1” Ação: “F+”

A eletroválvula “F+” é ativada para permitir o embolo do cilindro pneumático da mesa indexadora avançar e girar a mesa em 90 graus.

Transição: “GiraF1” → “GiraF2” Lógica: “$\overline{F1}$”

O sensor “F1” torna-se falso, mas o embolo do cilindro F ainda não girou a mesa indexadora. O estado “GiraF1” é desativado e o estado “GiraF2” é ativado.

Estado: “GiraF2” Ação: “F+”

A eletroválvula “F+” é ativada para permitir o embolo do cilindro pneumático da mesa indexadora avançar e girar a mesa em 90 graus.

Transição: “GiraF2” → “TempoF” Lógica: “F1”

O sensor “F1” torna-se verdadeiro, e o embolo do cilindro F girou a mesa indexadora em 90 graus. O estado “GiraF2” é desativado e o estado “TempoF” é ativado.

Estado: “TempoF” Ação: “temporizador T3 de 2 seg.”

O controlador espera o tempo de 2 segundos para garantir o sincronismo entre o término do giro da mesa indexadora e o avanço do embolo do cilindro A no programa 1.

Transição: “TempoF” → “EsperaF” Lógica: “F1.T3”

A mesa indexadora girou 90 graus e o temporizador de 2 terminou a contagem. O estado “TempoF” é desativado e o estado “EsperaF” é ativado.

Codificação do programa em linguagem Ladder1

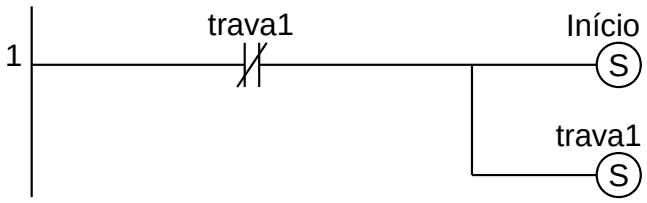

Figura 7.102 – Programa em linguagem Ladder para a inicialização da parte 1.

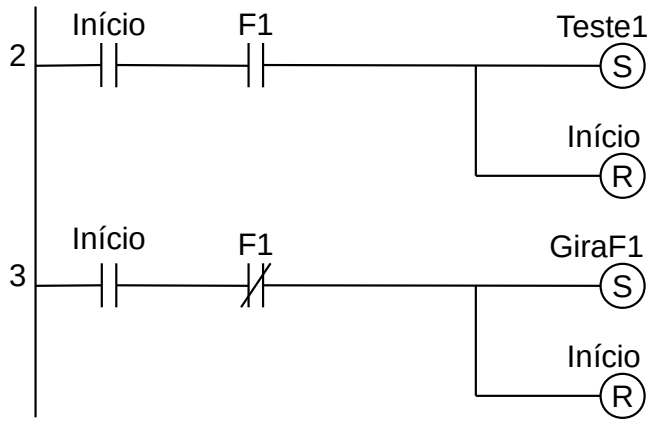

Figura 7.103 – Programa em linguagem Ladder para as transições saintes do estado Início.

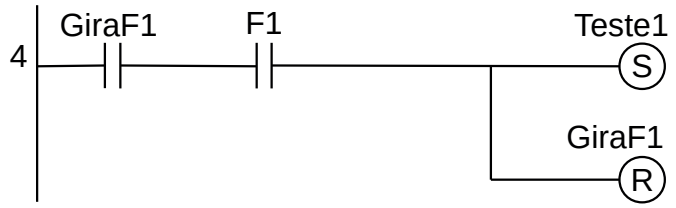

Figura 7.104 – Programa em linguagem Ladder para as transições saintes do estado GiraF1.

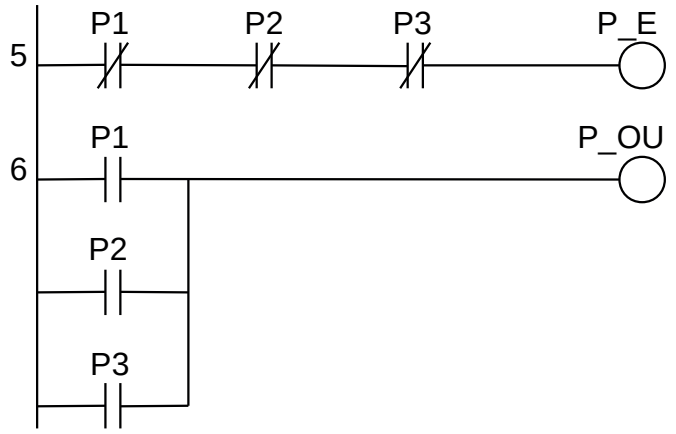

Figura 7.105 – Programa em linguagem Ladder para as frases lógicas do Teste1.

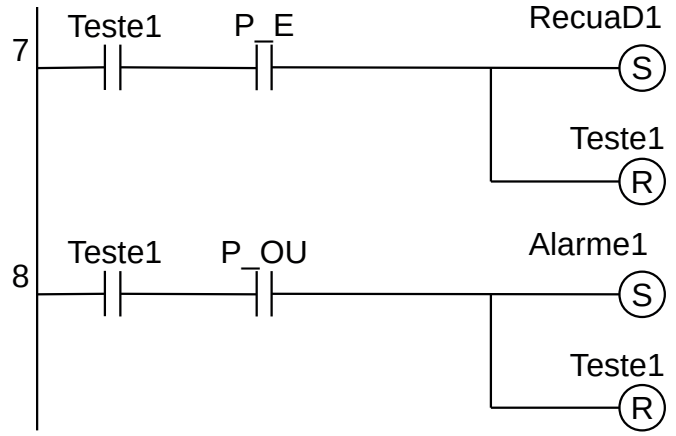

Figura 7.106 – Programa em linguagem Ladder para as transições saintes do estado Teste1.

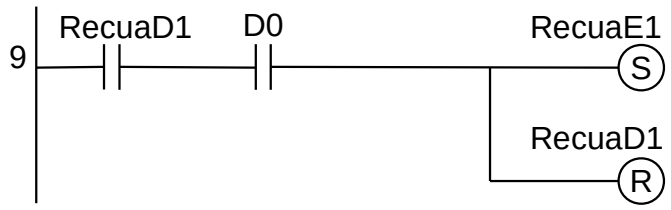

Figura 7.107 – Programa em linguagem Ladder para as transições saintes do estado Recua_D1.

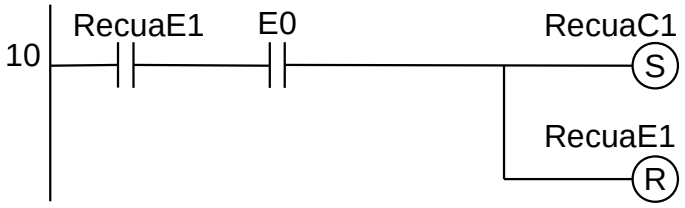

Figura 7.108 – Programa em linguagem Ladder para as transições saintes do estado RecuaE1.

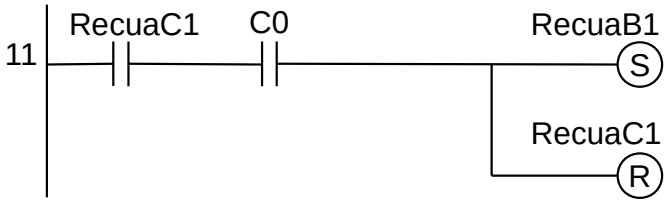

Figura 7.109 – Programa em linguagem Ladder para as transições saintes do estado RecuaC1.

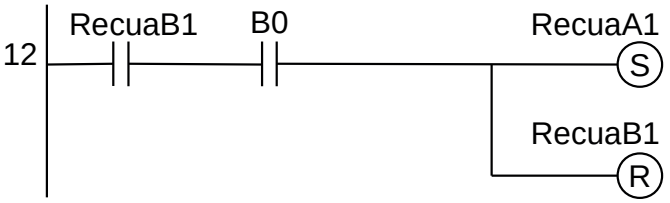

Figura 7.110 – Programa em linguagem Ladder para as transições saintes do estado RecuaB1.

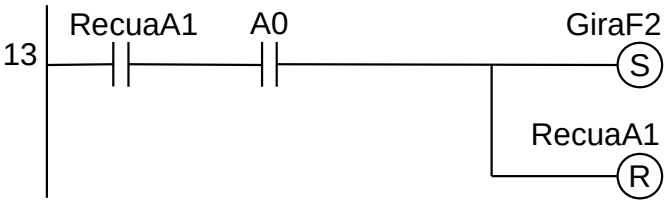

Figura 7.111 – Programa em linguagem Ladder para as transições saintes do estado RecuaA1.

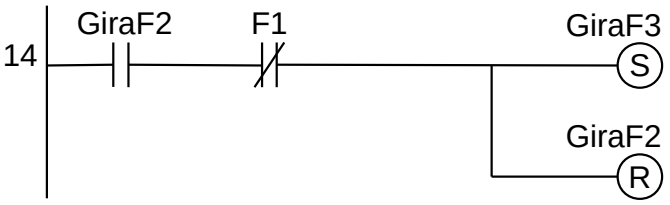

Figura 7.112 – Programa em linguagem Ladder para as transições saintes do estado GiraF2.

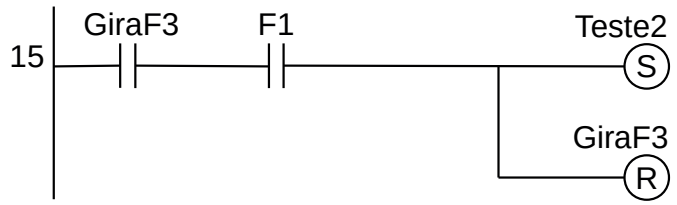

Figura 7.113 – Programa em linguagem Ladder para as transições saintes do estado GiraF3.

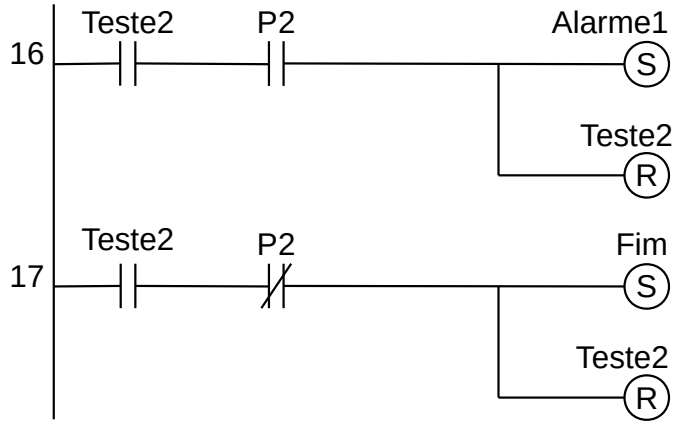

Figura 7.114 – Programa em linguagem Ladder para as transições saintes do estado Teste2.

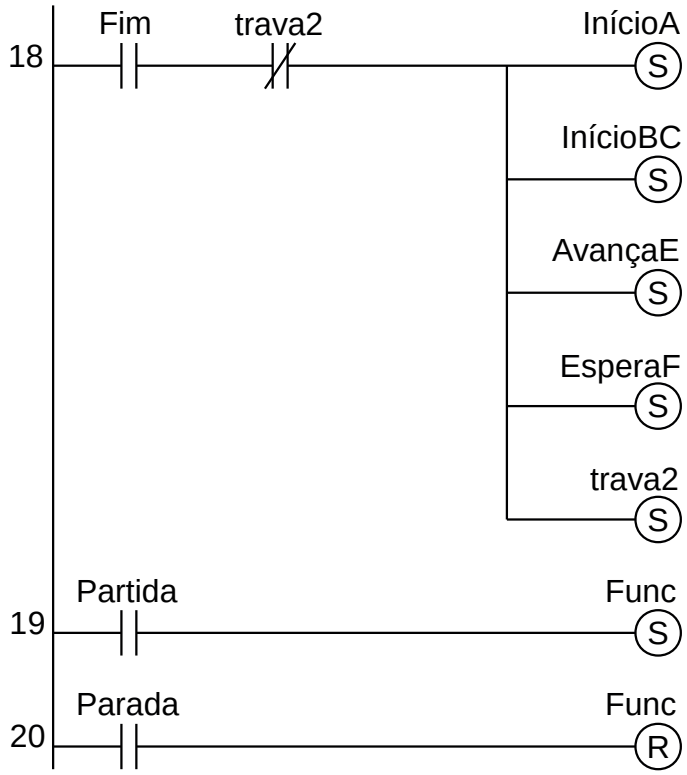

Figura 7.115 – Programa em linguagem Ladder para a inicialização da parte 2.

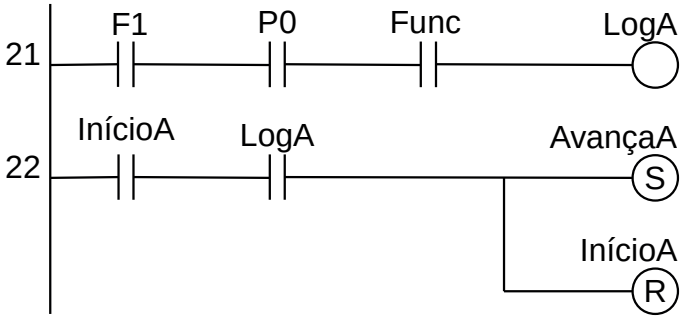

Figura 7.116 – Programa em linguagem Ladder para as transições saintes do estado InícioA.

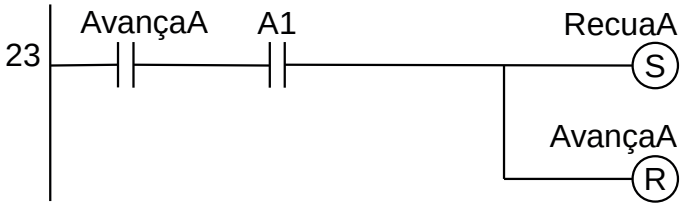

Figura 7.117 – Programa em linguagem Ladder para as transições saintes do estado AvançaA.

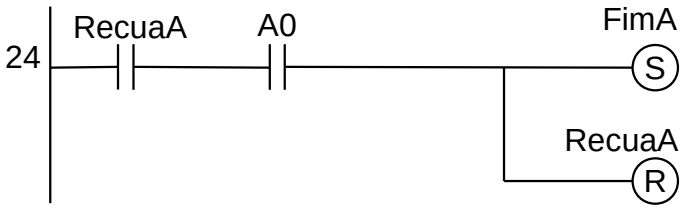

Figura 7.118 – Programa em linguagem Ladder para as transições saintes do estado RecuaA.

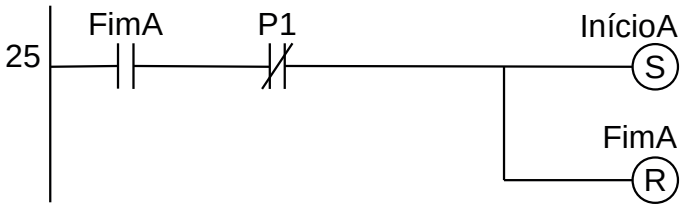

Figura 7.119 – Programa em linguagem Ladder para as transições saintes do estado FimA.

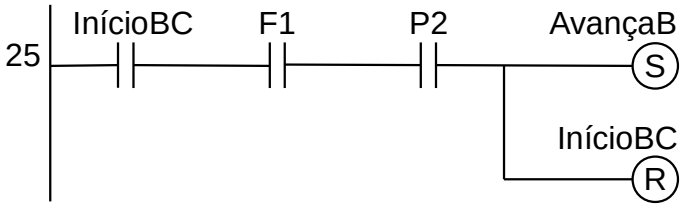

Figura 7.120 – Programa em linguagem Ladder para as transições saintes do estado InícioBC.

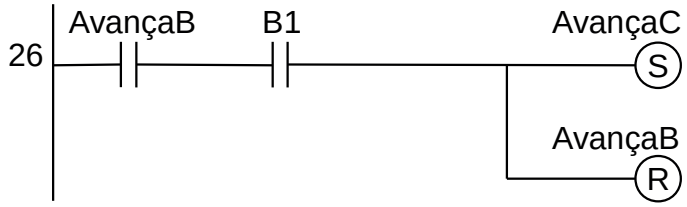

Figura 7.121 – Programa em linguagem Ladder para as transições saintes do estado AvançaB.

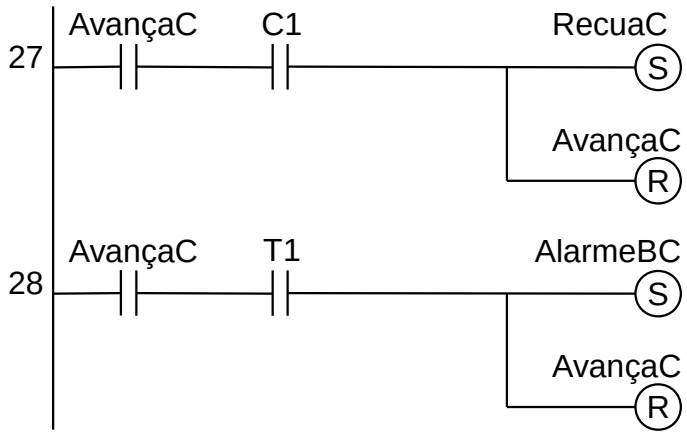

Figura 7.122 – Programa em linguagem Ladder para as transições saintes do estado AvançaC.

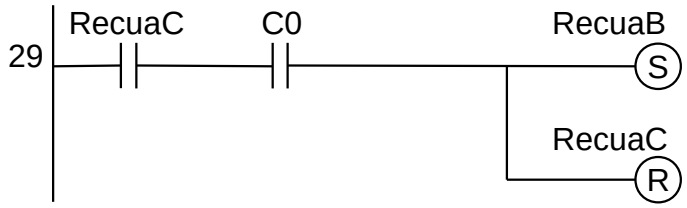

Figura 7.123 – Programa em linguagem Ladder para as transições saintes do estado RecuaC.

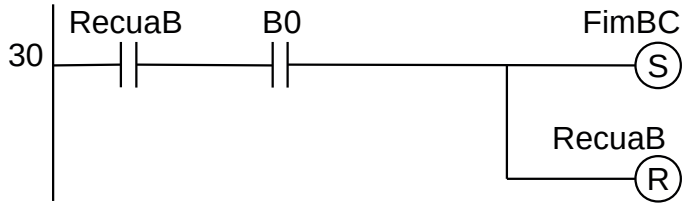

Figura 7.124 – Programa em linguagem Ladder para as transições saintes do estado RecuaB.

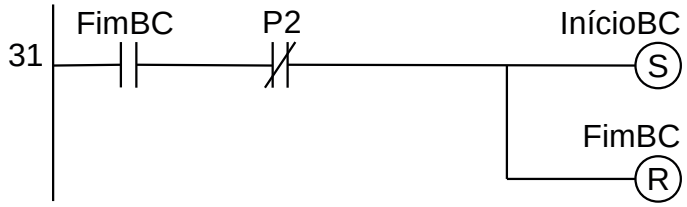

Figura 7.125 – Programa em linguagem Ladder para as transições saintes do estado FimBC.

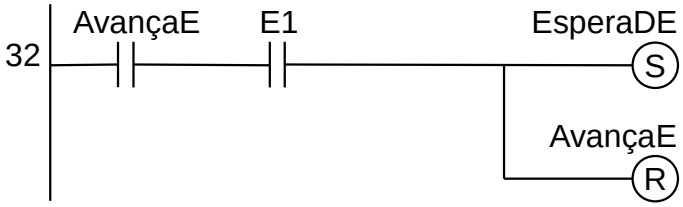

Figura 7.126 – Programa em linguagem Ladder para as transições saintes do estado AvançaE.

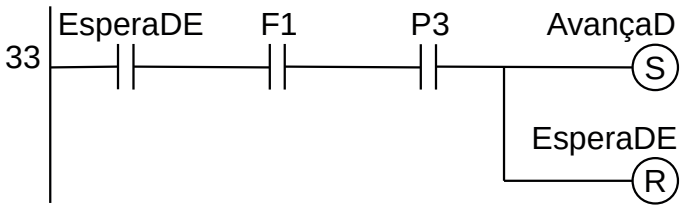

Figura 7.127 – Programa em linguagem Ladder para as transições saintes do estado EsperaDE.

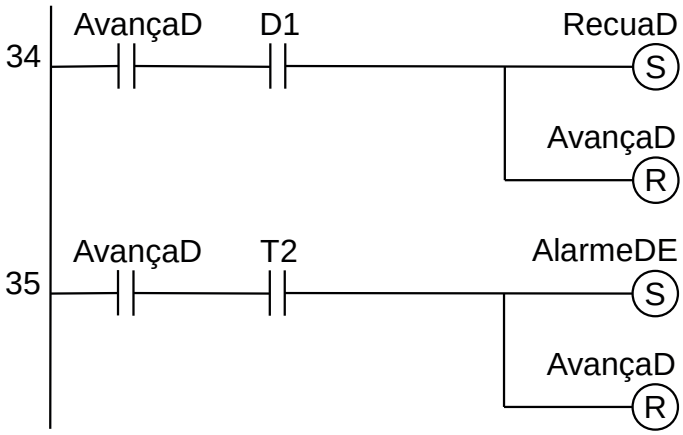

Figura 7.128 – Programa em linguagem Ladder para as transições saintes do estado AvançaD.

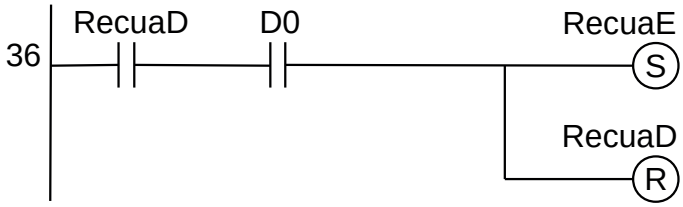

Figura 7.129 – Programa em linguagem Ladder para as transições saintes do estado RecuaD.

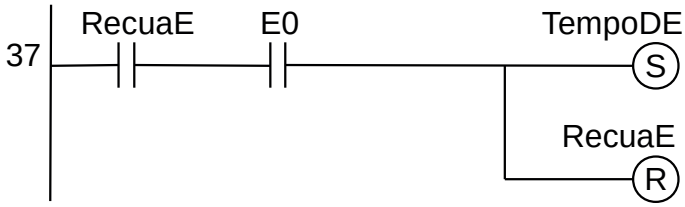

Figura 7.130 – Programa em linguagem Ladder para as transições saintes do estado RecuaE.

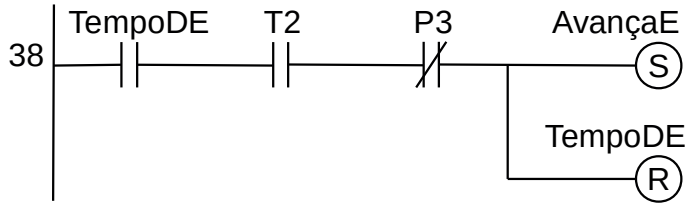

Figura 7.131 – Programa em linguagem Ladder para as transições saintes do estado TempoDE

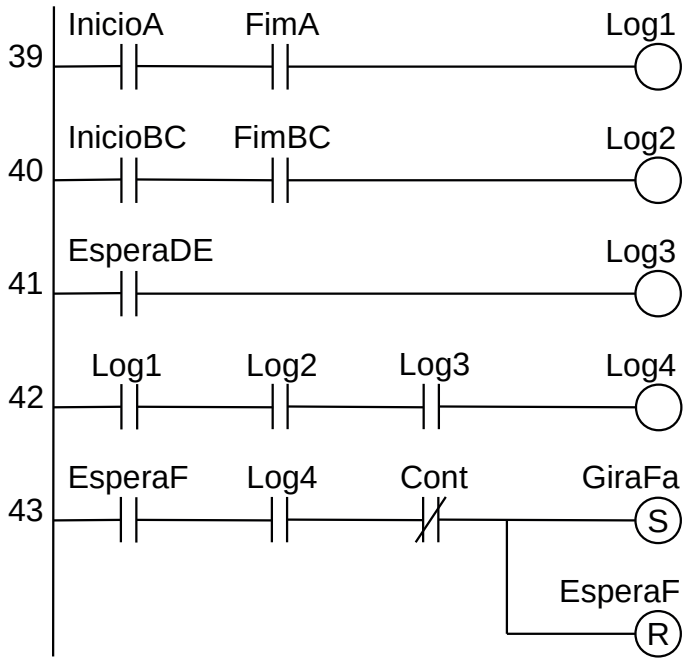

Figura 7.132 – Programa em linguagem Ladder para as transições saintes do estado EsperaF.

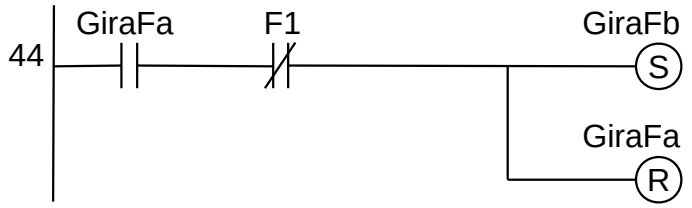

Figura 7.133 – Programa em linguagem Ladder para as transições saintes do estado GiraFa.

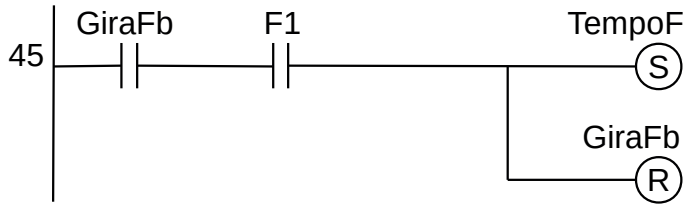

Figura 7.134 – Programa em linguagem Ladder para as transições saintes do estado GiraFb.

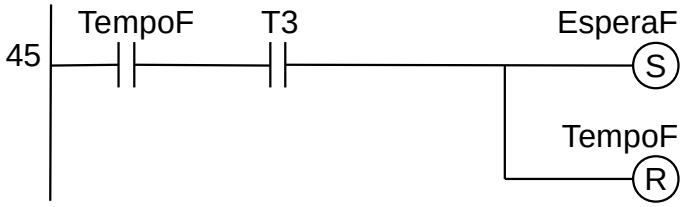

Figura 7.135 – Programa em linguagem Ladder para as transições saintes do estado TempoF.

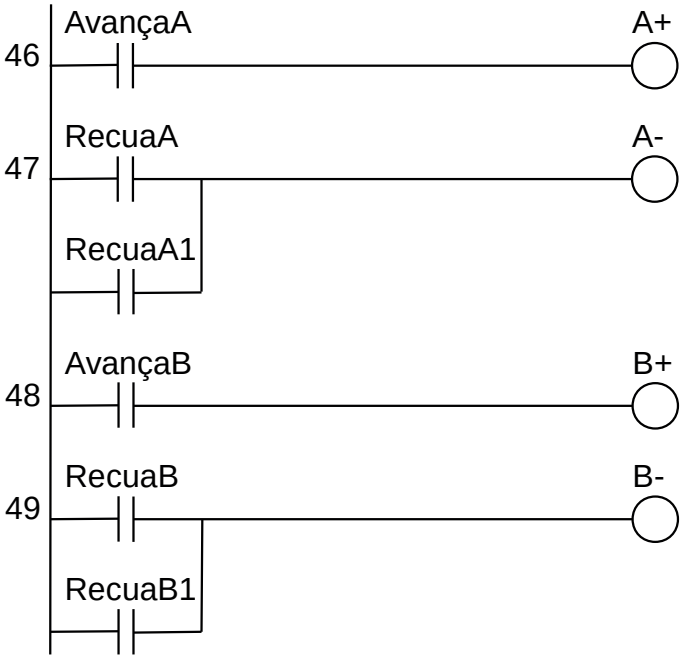

Figura 7.136 – Programa em linguagem Ladder para ativação das saídas (parte 1).

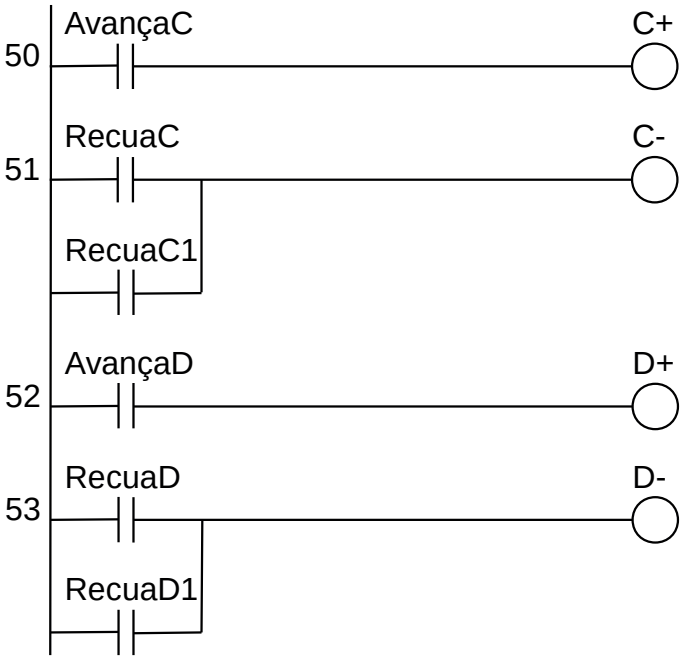

Figura 7.137 – Programa em linguagem Ladder para ativação das saídas (parte 2).

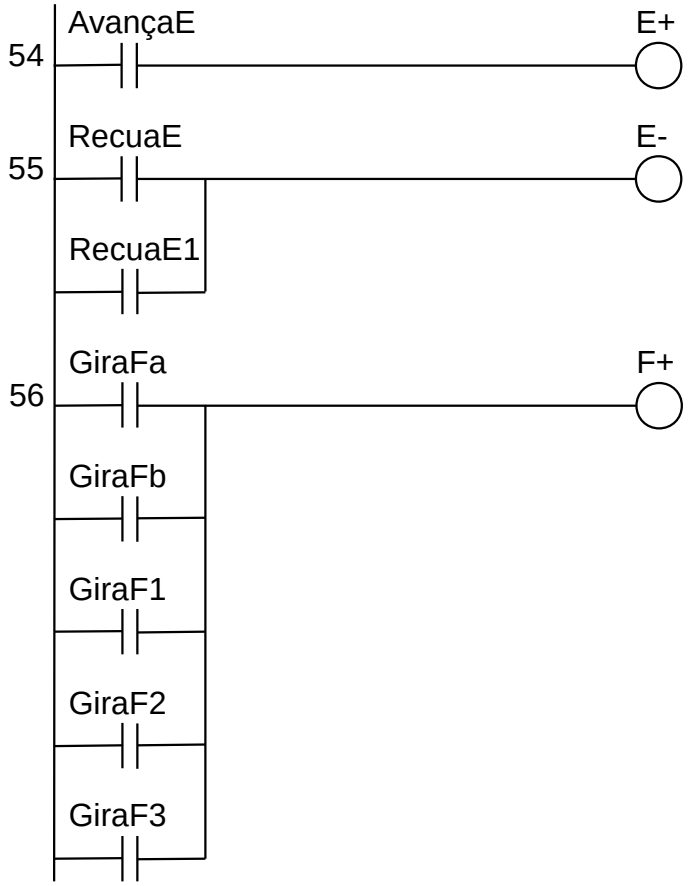

Figura 7.138 – Programa em linguagem Ladder para ativação das saídas (parte 3).

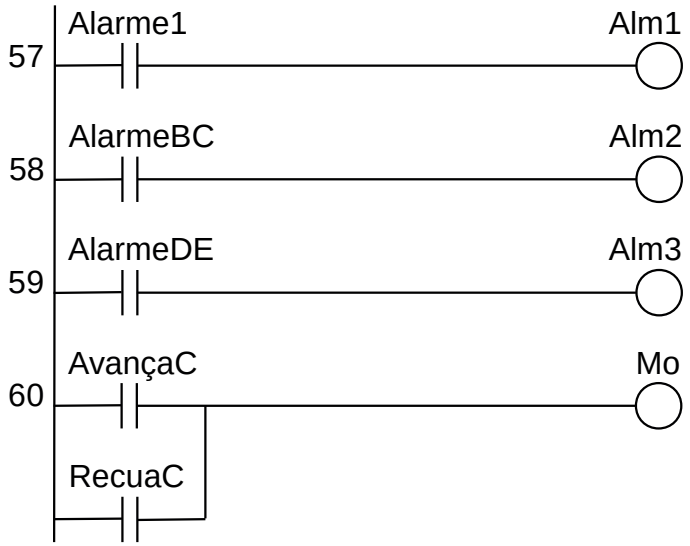

Figura 7.139 – Programa em linguagem Ladder para ativação das saídas (parte 4).

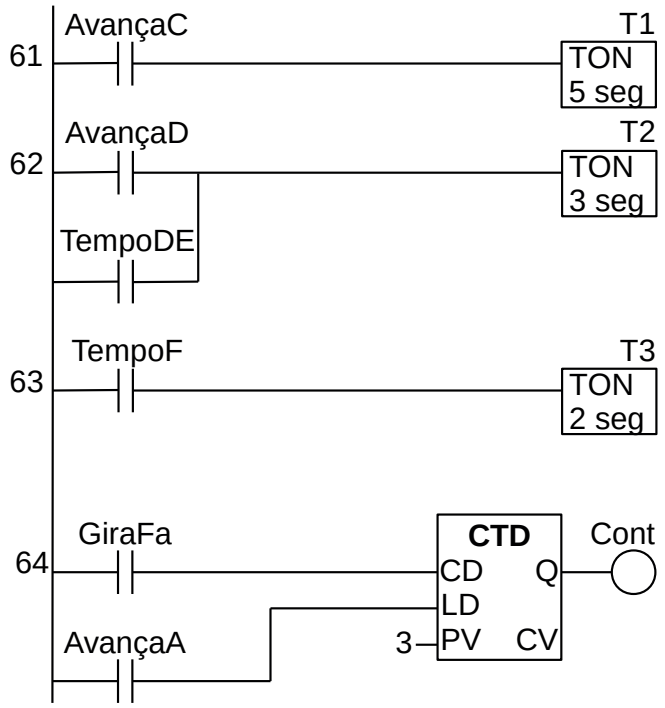

Figura 7.140 – Programa em linguagem Ladder para ativação das saídas (parte 5).

Para finalizar a programação é aconselhável a montagem de uma tabela com as etiquetas e descrição de cada variável interna, semelhante às tabelas 7.17 e 7.18. As tabelas com as variáveis internas podem ser vistas nas tabelas 7.19.

Variável	Etiqueta	Descrição
M01	Início	Memória para estado.
M02	GiraF1	Memória para estado.
M03	Teste1	Memória para estado.
M04	Alarme1	Memória para estado.
M05	RecuaD1	Memória para estado.
M06	RecuaE1	Memória para estado.
M07	RecuaC1	Memória para estado.
M08	RecuaB1	Memória para estado.
M09	RecuaA1	Memória para estado.
M10	GiraF2	Memória para estado.
M11	GiraF3	Memória para estado.
M12	Teste2	Memória para estado.
M13	Fim	Memória para estado.
M14	InícioA	Memória para estado.
M15	AvançaA	Memória para estado.
M16	RecuaA	Memória para estado.
M17	FimA	Memória para estado.
M18	InícioBC	Memória para estado.
M19	AvançaB	Memória para estado.
M20	AvançaC	Memória para estado.

Tabela 7.19 – Descrição das variáveis internas (parte 1).

Variável	Etiqueta	Descrição
M21	RecuaC	Memória para estado.
M22	RecuaB	Memória para estado.
M23	FimBC	Memória para estado.
M24	AlarmeBC	Memória para estado.
M25	AvançaE	Memória para estado.
M26	EsperaDE	Memória para estado.
M27	AvançaD	Memória para estado.
M28	RecuaD	Memória para estado.
M29	RecuaE	Memória para estado.
M30	TempoDE	Memória para estado.
M31	AlarmeDE	Memória para estado.
M32	EsperaF	Memória para estado.
M33	GiraFa	Memória para estado.
M34	GiraFb	Memória para estado.
M35	TempoF	Memória para estado.
M36	P_E	Memória interna temporária.
M37	P_OU	Memória interna temporária.
M38	LogA	Memória interna temporária.
M39	Log1	Memória interna temporária.
M40	Log2	Memória interna temporária.
M41	Log3	Memória interna temporária.
M42	Log4	Memória interna temporária.
M43	Func	Memória de funcionamento do programa.
M44	trava1	Memória de trava de inicialização.
M45	trava2	Memória de trava de inicialização.
T01	T1	Temporizador para ligar (TON) 5 segundos.
T02	T2	Temporizador para ligar (TON) 3 segundos.
T03	T3	Temporizador para ligar (TON) 2 segundos.
C01	Cont	Contador regressivo (PV=3).

Tabela 7.19 – Descrição das variáveis internas (parte 2).

7.6 – Conclusão

Uma abordagem que deve ser feita ao terminar um diagrama de estados é proceder a um conjunto de testes para verificar algumas condições de validação do diagrama realizado.

Esses testes podem ser classificados em:

Sintaxe

"O diagrama segue as regras?"

Especialista no domínio

"O diagrama está correto?"

"O que mais há que não está descrito neste diagrama?"

Rastreabilidade

"Tudo neste diagrama remonta correta e completamente ao seu predecessor?"

"Está tudo no predecessor refletido completa e corretamente neste diagrama?"

Diagramas de estado

Os diagramas de estado descrevem todos os estados que um objeto pode ter, os eventos sob os quais um objeto muda de estado (transições), as condições que devem ser atendidas antes que a transição ocorra (guardas) e as atividades realizadas durante a vida de um objeto (ações). Os diagramas de estado são muito úteis para descrever o comportamento de objetos individuais em todo o conjunto de casos de uso que afetam esses objetos. Os diagramas de estado não são úteis para descrever a colaboração entre objetos que causam as transições.

Notação

Estado. Uma condição durante a vida de um objeto em que ele satisfaz alguma condição, executa alguma ação ou espera por algum evento.

Evento. Uma ocorrência que pode desencadear uma transição de estado. Os tipos de eventos incluem um sinal explícito de fora do sistema, uma invocação de dentro do sistema, a passagem de um período de tempo designado ou uma condição designada se tornando verdadeira.

Guarda. Uma expressão booleana que, se verdadeira, permite que um evento cause uma transição.

Transição. A mudança de estado dentro de um objeto.

Ação. Uma ou mais ações executadas por um objeto em resposta a uma mudança de estado.

Teste de sintaxe

O tipo mais simples de teste é o teste de sintaxe. Ao realizar o teste de sintaxe, verificamos se o diagrama de estado contém informações corretas e adequadas. As perguntas que devem ser verificadas são:

- Cada diagrama de transição de estado tem um e apenas um estado inicial?
- Se o diagrama de transição de estado for em malha aberta, há, pelo menos, um estado terminal?
- Se o diagrama de estado é um circuito fechado, é realmente? (Quase todos são realmente de malha aberta)
- Cada estado tem, pelo menos, uma transição de saída? (Estados terminais não tem!)
- Se houver vários guardas para um único evento, os guardas são mutuamente exclusivos?
- Cada estado tem exatamente uma transição para cada combinação possível de guarda de eventos?
- Todos os estados ou transições redundantes ou duplicados foram removidos?
- Todos os estados são acessíveis?
- Cada estado "real" no mundo é representado por um e apenas um estado no diagrama?
- Cada estado e transição são claramente nomeados?
- Todos os caminhos possíveis também são caminhos válidos?
- Todos os caminhos válidos estão representados?

Teste especialista de domínio

Depois de verificar a sintaxe dos diagramas de estado, passamos ao segundo tipo de teste de especialista de domínio. Fazemos três tipos de perguntas: Está completa? Está correto? É consistente?

Completo:

- Todos os estados, eventos, protetores, transições e ações necessários são mostrados no diagrama?
- Todos os casos excepcionais são tratados de forma adequada?

Correto:

- Estamos usando diagramas de estado apenas para classes que têm um comportamento complexo e interessante?
- O diagrama representa corretamente a natureza de malha aberta / malha fechada da classe?
- Todos os estados, eventos, guardas, transições e ações necessários estão definidos corretamente?

Consistente:

- Existe uma correspondência um a um entre os eventos de um objeto e seus métodos?

Teste de Rastreabilidade

Queremos ter certeza de que podemos rastrear desde os requisitos até os diagramas de estado e dos diagramas de estado de volta aos requisitos. Novamente, nos voltamos para uma questão: é consistente?

Consistente:

- Todos os estados, eventos, proteções, transições e ações nos requisitos aparecem no diagrama de transição de estados?

ISBN 978-655668037-8

9 786556 680378