Parte 01 – Introdução à Automação & Controle

1.0 Introdução

A automação industrial provoca uma verdadeira revolução no ambiente industrial. A necessidade de um profissional altamente qualificado é imprescindível para o sucesso da mesma. Hoje, o profissional deve ser o mais versátil possível, conhecer o máximo de todas as áreas.

Na automação, o profissional mais desejado é o que conhece eletrônica, mecânica, programação e que consiga integrar tudo em um projeto final que atenda as necessidades da empresa. Este profissional é caro e difícil de encontrar, este é o perfil do profissional de automação e controle.

Este trabalho se propõe: auxiliar projetistas em uma das áreas mais carentes do mercado, a automação e controle. Com foco principal na elaboração de programas desenvolvidos para CLP (**C**ontroladores **L**ógicos **P**rogramáveis), que tem sido o grande problema, pois são feitos empiricamente, ou seja, sem lógica, o que ocasiona em programas longos, levando assim o projetista/ programador a um desgaste desnecessário.

A utilização de técnicas especificas para programação é sem dúvida a forma mais rápida e simples de programar um CLP. Neste trabalho, propomos técnicas bastante conhecidas em outras áreas, porém pouco utilizadas na programação de CLP. São elas:

- **Lógica combinacional simples** ⇒ São blocos que executam funções lógicas simples, como AND, OR, NOT, NAND, NOR etc. A maioria dos controladores disponibilizam estes blocos prontos, bastando ao projetista implementá-las ao sistema.

- **Mapas de Veitch-Karnaugh** (Veitch-Karnaugh Map)⇒ técnica utilizada para simplificação dos circuitos lógicos, muito utilizada em eletrônica digital, aqui será adaptada para solução de sistemas que necessitam de lógica de dificuldade média.

- **Máquina de estado Finito** (Finite State Machine)⇒ também conhecida como diagrama de estado, são largamente utilizados para modelar o comportamento de aplicativos como: projeto de hardware de sistemas digitais, engenharia de software, no estudo da computação e das linguagens. Este tipo de técnica será adaptado para modelagem de sistemas de automação que envolve um pouco mais de dificuldade.

- **Algoritmos computacionais**⇒ São utilizados em soluções extremamente complexas, estes necessitam de softwares adequados e não serão discutidos neste trabalho.

Precisamos agora, conhecer um pouco de sobre automação e controle e seu principal dispositivo controlador o CLP, antes iniciarmos, as técnicas de programação.

1.1 O que é automação?

Na indústria moderna as mudanças são muito rápidas, hoje se fabrica um produto “X” amanhã a ordem é fabricar o produto “Y”. Antigamente, todo um hardware utilizado para produção do produto “X” era substituído ou readequado para produzir o produto “Y”. Hoje, isso não é mais possível, o tempo para adequar à linha para um novo produto é convertido em prejuízo, pois a linha fica parada.

Linhas de montagens longas, com vários funcionários, executando trabalhos repetitivos levam empresas à um alto custo operacional, o que não é interessante.

Com o advento automação, a indústria não precisa mais de mão de obra, trabalho muito valorizado no passado, mas em extinção nos dias atuais. Isto porque, a mão de obra é substituída por máquinas e equipamentos capazes de realizar as mesmas funções dos operários, com maior velocidade e qualidade superior.

1.2 Automação versus mecanização

O simples uso de máquinas para substituir o trabalho manual, não é automação, isto é pura mecanização. Devemos tomar alguns cuidados no que diz respeito ao conceito de automação. Por exemplo, para que um processo industrial seja automatizado deve haver a interação de três agentes.

- ✓ Processo flexível ⇒ Quanto mais flexível é a produção, maior o grau de inteligência da automação.
- ✓ Sistema de controle ⇒ É formado por uma série de dispositivos elétricos, eletrônicos, capazes de gerenciar o comportamento de um processo industrial.
- ✓ Sistema mecanizado ⇒ É “fácil” projetar um braço mecânico capaz de pegar uma peça de uma posição X e levar posição Y, o difícil é fazer o braço seja inteligente e perceba a peça, verifique o seu tamanho, coloque-a na posição adequada é um pouco mais complicado, mas isto sim é automação.

Assim, se tivermos um sistema mecanizado, e ele atribuirmos um sistema de controle ou inteligência teremos então, um sistema automatizado.

1.3 A automação na indústria?

Atender as necessidades dos clientes é o grande objetivo da indústria. O cliente não quer mais um produto feito de qualquer jeito, ele quer qualidade, durabilidade, que atenda todas as suas necessidades e o principal a baixo custo. Assim, a indústria tem que se adequar modernizar suas máquinas, equipamentos e principalmente contratar profissionais capazes de atender a necessidade de um mercado cada vez mais exigente. Vale ressaltar, que a indústria seja pequena, média ou de grande porte que não se adequar está fadada ao fracasso.

1.4 Automação versus emprego.

Dizer que a automação esta acabando com os empregos é um erro, nunca houve tantas ofertas de emprego. É verdade sim, que muitas frentes de trabalho acabaram, muitas profissões foram extintas, porém, muitas outras surgiram.

Também é verdade dizer que uma linha montagem automatizada o robô tira emprego no montador, porém, quem monta a linha, quem faz as peças para esta linha, quem projeta a linha, quem programa a linha?

O que não existe mais é oferta de mão de obra que não exige qualificação, o dito "peão". O funcionário que não se especializar está fadado ao fracasso profissional.

1.5 A engenharia de automação e controle - mecatrônica

O Comitê Assessor para Pesquisa e Desenvolvimento Industrial da Comunidade Européia (IRDAC) define mecatrônica como:

"A integração sinérgética da engenharia mecânica com a eletrônica e o controle inteligente por computador no projeto de processos e de manufatura de produtos".

Em outras palavras podemos dizer que a mecatrônica é a junção das engenharias mecânica, elétrica e eletrônica, aliada a avançadas técnicas de computacionais para aquisição e processamento de dados.

Mecatrônica é o curso da moda, sem dúvida, o grande filão do mercado educacional, uma das áreas mais novas da engenharia em todo o mundo. Porém, poucos estão preparados para receberem tamanha bagagem intelectual, em tão pouco tempo.

Tudo isso torna o curso de engenharia mecatrônica mais interessante e extremamente desafiador, uma vez que poucos atuarão nesta área.

1.6 Máquinas x homem

O homem é dotado de capacidades físicas fascinantes como andar, correr, nadar etc. Assim, ele pode fazer o que desejar, ou julgar necessário, desde que dentro de suas limitações. Cabe salientar que, as limitações do homem estão na parte física, e não na intelectual, por exemplo, não se consegue levantar uma tonelada, porém, consegue projetar uma máquina para levanta-la. A "capacidade física" das máquinas é o que chamamos de hardware e quem diz o que é necessário é o software.

- ✓ O Hardware é constituído pelas partes mecânicas e eletrônicas de um projeto de automação e controle. Entre os mais comuns estão:

- Controladores lógicos programáveis;
- Microcontroladores e microprocessadores;
- Estrutura física de redes de comunicação;
- Sensores e transdutores;
- Atuadores;
- Interfaces homem máquina (IHM);
- Periféricos.

✓ O software é o programa computacional implementado no sistema automatizado. É o que vai dizer ao hardware o que fazer. Os mais comuns são:

- Algoritmos de controle;
- Sistemas de comunicação de dados;
- Sistemas supervisórios.

O grande objetivo deste trabalho está em relacionar todos estes itens de Hardware e Software ao carro chefe da automação e Controlador Lógico Programável.

"Escolha o trabalho de que gostas e não terás de trabalhar um único dia em tua vida".
Confúcio

www.clubedaeletronica.com.br

Referências bibliográficas:

- http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/
- http://www.cpdee.ufmg.br/~carmela/NORMA%20IEC%201131.doc
- http://www.software.rockwell.com/corporate/reference/Iec1131/
- http://www.plcopen.org/
- http://www.lme.usp.br/~fonseca/psi2562%20aula%206%20IHM.pdf
- http://www.teses.usp.br/teses/disponiveis/18/18133/tde-11072002-085859/
- http://www.redenet.edu.br/publicacoes/arquivos/20080108_144615_INDU-058.pdf
- http://www.corradi.junior.nom.br/modCLP.pdf
- http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/

Parte 02 – O Controlador Lógico Programável

2.0 Introdução:

Para controlar uma planta industrial, seja a mais simples ou complexa, necessitamos de um sistema de controle, obviamente que quanto mais complexa mais se exige deste controle. Porém, a idéia do sistema de controle é bastante simples e pode ser dividida em três partes:

- **Entradas**, que são responsáveis pela coleta de informações;
- **Dispositivo controlador**, que manipula as informações através de um software previamente embutido; e
- **Saídas** que são responsáveis pela ação.

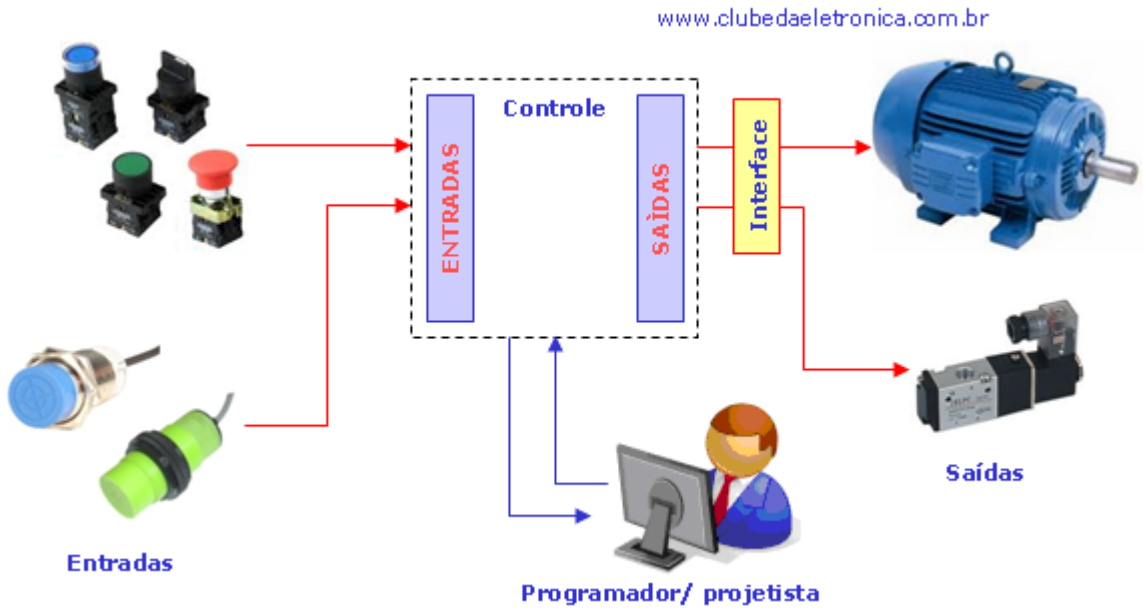


Figura 2.1 - Sistema de controle

Descrição de cada bloco:

- **Entradas**: São os sensores analógicos ou digitais. São os responsáveis por alimentar o controle com as informações externas.

- Os sensores digitais captam e enviam aos controladores variáveis discretas, tais como ligado/desligado ou alto/baixo, enquanto os analógicos captam e enviam aos controladores variáveis com uma faixa contínua, tais como pressão, temperatura, vazão ou nível.

- **Controlador**: Recebe informações do estado o estado real da planta através de uma rede de sensores ligados a ele, então ele compara com um algoritmo de controle, previamente embutido no controlador, que atualiza suas saídas.

- **Saídas**: São os atuadores, dispositivos com finalidade de provocar uma ação. Os atuadores mais comuns são as válvulas e os motores.

O operador através de um sistema de supervisão interage com o controlador e consequentemente com a planta através dos parâmetros de controle. O sistema de supervisão monitora todo o processo gerando relatórios e gráficos de tendência extremamente úteis para melhorias do processo industrial.

2.1 Controlador lógico programável

É o controlador mais utilizado em sistemas de controle e, portanto, na automação além de ser a ideia central deste trabalho. Também conhecido como CLP (**C**ontrolador **L**ógico **P**rogramável) ou PLC (**P**rogrammable **L**ogic **C**ontroller). Sua função como o próprio nome diz é controlar a planta ou processo através de um algoritmo lógico programado. Desse modo, sinais de entrada provenientes de sensores são logicamente combinados gerando sinais de saída para os atuadores. De maneira simplificada, podemos dizer que o CLP é o cérebro da automação e controle.

Antes, quando um operário novo era contratado para uma linha de montagem, alguém que já conhecia o serviço o instruía, ou seja, dizia a ele tudo o que ele devia fazer, ele armazenava as informações e realizava o trabalho facilmente. Hoje, não é diferente, porém, o "operário" é uma máquina automatizada e quem tem que dizer à ela o que fazer é o programador. No âmbito industrial, quem recebe as instruções é o CLP, e este, assim como o operário se não for bem instruído, não executará bem sua função.

2.2 Definições de CLP

A ABNT (Associação Brasileira de Normas Técnicas) define o CLP como:

"Um equipamento eletrônico digital com hardware e software compatíveis com aplicações industriais"

A NEMA (National Electrical Manufactures Association), diz que:

"É um aparelho eletrônico digital que utiliza uma memória programável para armazenar internamente instruções e para implementar funções específicas, tais como lógica seqüencial, temporização, contagem e aritmética, controlando, por meio de módulos de entradas e saídas, vários tipos de máquinas ou processos".

2.3 Breve histórico

No passado, mudar uma linha de montagem demorava muito tempo, e tempo é dinheiro. A necessidade de mudança constante em linhas de produção obrigou a indústria, em especial a automobilística a procurar uma solução viável para este problema.

Foi em 1969, que sob a liderança do engenheiro Richard Morley, foi introduzido na linha de montagem da General Motor nos EUA um dispositivo capaz de sequenciador estados.

Na década de 70, graças a evolução da eletrônica digital, o CLP ganhou alguns blocos com funções prontas, como temporizadores, contadores, movimentação de dados, controle de PID entre outros, o que facilitou bastante sua programação.

Nascia assim, um equipamento bastante versátil e de fácil utilização, que vem se aprimorando constantemente, diversificando e modernizando cada vez mais os processos industriais.

2.4 Princípio de funcionamento

O CLP funciona de forma sequencial, ou seja, faz um ciclo de varredura em seus estados (chamasse de estado as etapas do processo). É importante observar que quando cada estado do ciclo é executado, os outros ficam inativos. O tempo total para realizar o ciclo é denominado clock. Isso justifica a exigência de processadores com velocidades cada vez mais altas. O diagrama de blocos abaixo ilustra de forma simplificada o funcionamento do CLP.

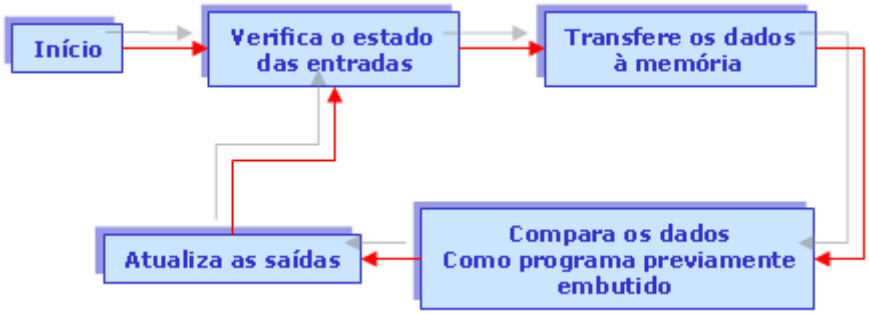


Figura 2.2 – Princípio de funcionamento do CLP

Descrição de cada bloco:

- **Início:** Verifica o funcionamento do controlador CPU, memórias, circuitos auxiliares, estado das chaves, existência de um programa de usuário e emite aviso de erro em caso de falha. Desativa todas as saídas.

- **Verifica o estado das entradas:** Lê cada uma das entradas, verificando se houve acionamento. O processo é chamado de ciclo de varredura.

- **Compara com o programa do usuário:** Compara as entradas com um algoritmo embutido e envia as informações às saídas.

- **Atualiza as saídas:** As saídas são acionadas ou desativadas conforme a determinação do programa do usuário. Um novo ciclo é iniciado.

2.5 Principais características do CLP

Os controladores lógicos programáveis vieram para ficar, sua vantagem em relação ao outros controladores no ambiente industrial é inegável, e as razões para este sucesso são inúmeras, citaremos algumas:

- São robustos, construído para operar no chão de fábrica, área extremamente susceptível a interferências, local onde microcontroladores não são muitos bem quistos.

- Linguagem de programação gráfica, de alto nível, de fácil compreensão principalmente por conhecedores de esquemas elétricos.

- Reduz significativamente a fiação utilizada na instalação elétrica, barateando o custo de instalação.

- Modificações rápidas minimizam a possibilidade de erros, pois se muda a lógica, sem mudar a instalação o que muito útil devido à flexibilidade da indústria moderna.

- Possui uma infinidade de blocos prontos para o uso, evitando que o programador tenha de desenvolver algoritmos para funções como: temporizar, contar, calcular etc.

- Fácil comunicação permite, através de interfaces de operação que controladores e computadores troquem informações.

2.6 Arquitetura dos controladores lógicos programáveis

Os controladores lógicos são constituídos por fisicamente por entradas, processamento e saídas. Vejamos, a função de cada uma delas.

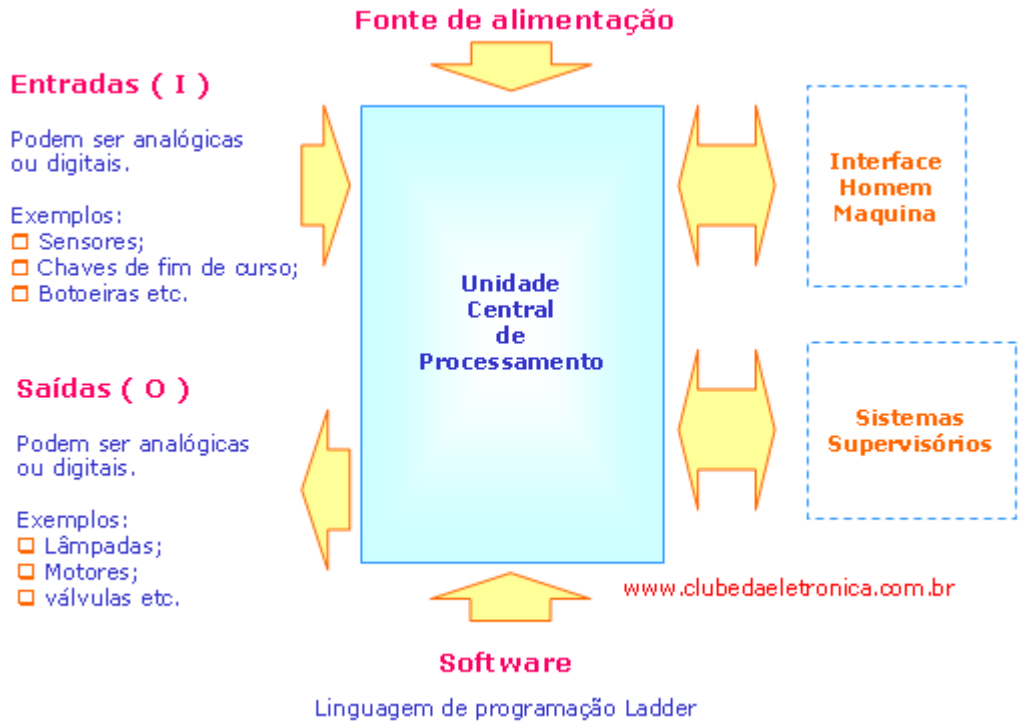


Figura 2.3 – Arquitetura básica do CLP

- **Entradas (I)** ⇒ Os controladores lógicos programáveis, assim como o cérebro humano, necessita receber informações de um determinado ambiente. Para receber estas informações estes equipamentos são dotados de entradas físicas que podem ser analógicas e/ou digitais.

- ◊ **Processamento** ⇒ Uma vez coletado os dados, estes devem ser processados a fim de que realizem uma determinada operação. Na unidade de processamento está gravado o firmware, que é o software do proprietário e controla diretamente o hardware.

- ◊ **Saídas (O)** ⇒ Uma vez que as informações foram coletadas e processadas, uma atitude deve ser tomada, ou seja, um sinal elétrico que pode ser analógico ou digital será enviado a saída.

Outros itens podem ser incorporados agregando valor ao sistema de controle são eles:

- ❑ ***Interface Homem Máquina (IHM)*** ⇒ É o elo de ligação entre o operador e a máquina, é a parte mais amigável, ou seja, de fácil uso permitido ao operador alterar entradas e/ou saídas do sistema ela ser local ou remota dependendo da necessidade.

- ❑ ***Sistemas supervisórios*** ⇒ É uma interface gráfica que permite ao usuário o coletar, monitorar e interagir com seu sistema em tempo real. É usado onde há necessidade contínua de:

  - Verificação do estado operacional da máquina;
  - Coletar valores de variáveis de processo;
  - Gerar relatórios e gráficos de tendência, etc.

Em tempo, vale lembrar que, definir bem o que são entradas e saídas é crucial para poder programar o CLP.

2.7 – Dispositivos de máquina

O CLP sozinho não faz nada, ele deve estar ligado a outros dispositivos capazes de coletar informações (sensores) e dispositivos capazes de atuar (atuadores) só assim sistema será completo.

2.7.1 – Dispositivos de entrada – Sensores

Hoje, dispositivos de alta tecnologia são capazes de substituir os sentidos humanos como, visão, audição, tato, olfato e paladar. Quanto maior a capacidade de sentir da automação, maior sua inteligência com isso mais independente do homem.

Alimentar as entradas do CLP com as informações coletadas pelos sensores e transdutores no ambiente é crucial, só assim o mesmo poderá processá-las. Os sensores analógicos ou digitais são sem dúvida o braço direito da automação, estes dispositivos que podem ser desde uma simples chave ao mais sofisticado sensor são ligados às entradas do CLP.

2.7.1.1- Entradas discretas e analógicas

Como o próprio nome diz é um controle digital, ou seja, controla eventos discretos, porém, se adicionados a ele conversores A/D e D/A, ele se torna um equipamento completo capaz de controlar tanto variáveis discretas como analógicas.

- ***Discretas***: São entradas que recebem informações em forma de pulsos elétricos "0" ou "1" não há um valor intermediário. Os sensores mais usados para esse fim são os:
  - **Mecânicos:** São sensores que operam de forma mecânica, ou seja, necessita contato. Não importa o material.
  - **Magnéticos:** São sensores que operam com campo magnético, detectam apenas magnetos.
  - **Indutivos:** São sensores que operam com campo eletro-magnético, portanto detectam apenas materiais ferromagnéticos.
  - **Capacitivos:** São sensores que operam com o principio de capacitância, detectam todos os tipos de materiais.
  - **Ópticos:** São sensores que operam com emissão de luz, estes detectam todos os tipos de materiais.
  - **Ultra-sônicos:** São sensores que operam com emissão e reflexão de um feixe de ondas acústicas. A saída comuta quando este feixe é refletido ou interrompido pelo material a ser detectado.
  - **Pneumáticos:** São sensores que se baseiam no desequilíbrio da pressão em uma determinada conexão do sensor. A saída comuta quando um jato de ar através do mesmo é alterado pela presença de um objeto.

- ***Analógicas***: Recebem informações de forma contínua, estas são convertidas em digitais para fins de processamento e enviadas às saídas. Os elementos responsáveis pelo envio desta informação ao CLP são dispositivos transdutores, ou seja, dispositivos que tem a finalidade de converter grandezas físicas em sinais elétricos. A maioria dos casos os sensores analógicos fornecem sinais de 0 a 10V, 0 a 5V ou 4 a 20mA, sendo o último o mais utilizado na automação. Assim, as maiorias dos controladores estão preparados para receber este tipo de sinal. Entre muitos transdutores, podemos citar:
  - **Strain gauge:** É um transdutor capaz de medir deformações de corpos. Quando um material é deformado sua resistência elétrica é alterada, a fração de mudança na resistência é proporcional a fração de mudança no comprimento do material.
  - **Termopar:** Possui uma junção de dois fios metálicos com coeficientes térmicos diferentes unidos. A tensão gerada é proporcional à diferença na temperatura entre estas junções.

2.7.2 - Dispositivo de Saída – Atuadores

É essencial em qualquer sistema, sua função é agir diretamente no processo, de acordo com sinais recebidos do controlador. A correta escolha do atuador depende do tipo de ação necessária, vejamos alguns atuadores:

- **Pneumáticos:** Um sistema pneumático é utilizado quando se precisa de força. Este tipo é normalmente empregado em sistemas onde se requer altas velocidades nos movimentos, com pouco controle sobre o posicionamento final, em aplicações onde o torque exigido é relativamente baixo. O acionamento é feito por intermédio de eletroválvulas que regulam o fluxo de ar no sistema. Em alguns processos contínuos, a válvula é capaz de regular a vazão do fluido (líquido, gás ou vapor) que passa pela tubulação.

- **Hidráulicos:** São utilizados em sistemas onde há a necessidade de maior torque, como é o caso de máquinas de grande porte, na forma de pistões ou motores hidráulicos. Apresentam alto desempenho e baixa manutenção. Assim como no caso dos pneumáticos, é comandado por eletroválvulas que controlam o fluxo de óleo do sistema.

- **Elétricos:** São largamente utilizados em aplicações industriais, como bombas, válvulas de controle, eixos de máquina-ferramenta, articulações de robôs, esteiras, entre outros. Como atuadores deste tipo estão os motores elétricos.

  - Um motor elétrico é um dispositivo que transforma energia elétrica em energia mecânica, em geral energia cinética, ou seja, a simples energia elétrica, seja contínua ou alternada, garante movimento em um eixo, que pode ser aproveitado de diversas maneiras dependendo da aplicação do motor. Entre os tipos de motor, destaca-se:

    a) **Motor de passo –** Utilizado em máquinas que necessitam de precisão nos movimentos. O motor de passo se comporta diferente de outros motores DC. Primeiramente ele não pode girar livremente quando alimentado, mas o faz em passos discretos. Para isto há a necessidade de um circuito lógico responsável em converter sinais de passo e de direção em comandos para os enrolamentos do motor.

    b) **Motor de corrente contínua –** É utilizado em máquinas que necessitam de velocidade e precisão, substituindo em muitos casos o motor de passo. Apresenta baixa relação peso/potência e baixo nível de ruído.

    c) **Motor de corrente alternada –** É utilizado onde se requer velocidade fixa e em altas potências, sua grande vantagem é que não necessita de drivers, adapta-se facilmente a qualquer máquina, além de baixo custo.

Os controladores lógicos são essenciais a lógica de controle, ele será o cérebro do sistema automatizado enquanto que as entradas serão seus sentidos e a saídas suas ações.

Em tempo, devemos lembrar que as entradas e saídas físicas do CLP não devem estar ligadas diretamente aos dispositivos de máquina, elas devem ser isoladas, ou seja, há necessidade de interfaces adequadas, para que não danifique o CLP, em caso de possíveis anomalias.

Uma vez conhecida a estrutura do CLP, deve-se conhecer a linguagens de programação, estas são normalizadas e são descritas em seguida.

A ação nem sempre traz felicidade, mas não há felicidade sem ação.

Benjamin Disraeli

www.clubedaeletronica.com.br

Referências bibliográficas:

- http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/
- http://www.cpdee.ufmg.br/~carmela/NORMA%20IEC%201131.doc
- http://www.software.rockwell.com/corporate/reference/Iec1131/
- http://www.plcopen.org/
- http://www.lme.usp.br/~fonseca/psi2562%20aula%206%20IHM.pdf
- http://www.teses.usp.br/teses/disponiveis/18/18133/tde-11072002-085859/
- http://www.redenet.edu.br/publicacoes/arquivos/20080108_144615_INDU-058.pdf
- http://www.corradi.junior.nom.br/modCLP.pdf
- http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/

Parte 03 - Linguagens para programação de CLP (Norma IEC 61131 – 3)

Introdução:

Há muito tempo se procura estabelecer um padrão para programação de CLP, em 1979, foi designado um grupo de trabalho com o IEC (International Electro-technical Comission) voltado para este propósito.

Este grupo tinha como objetivo analisar o projeto completo de CLP's (inclusive hardware), instalação, testes, documentação, programação e comunicações. Este grupo designou oito frentes de trabalho para desenvolver diferentes partes do padrão para CLP's.

Em 1992, o IEC publicou a norma IEC 1131, a qual estabelece padrões para Controladores Lógicos Programáveis. Em suas diversas versões a norma ganhou o número 6 passando assim, para IEC 61131. Esta está dividida em partes, que são:

- 61131-1 - Informações gerais
- 61131-2 - Requisitos de hardware
- 61131-3 - Linguagens de programação
- 61131-4 - Guia de orientação ao usuário
- 61131-5 – Comunicação

Outras três partes, ainda em fase de elaboração.

- 61131-6 - Comunicação via *Fieldbus*
- 61131-7 - Programação utilizando Lógica *Fuzzy*
- 61131-8 - Guia para implementação das linguagens

Em 1993, é publicada sua terceira parte a IEC 61131-3 estabelecendo um padrão global para programação de controladores lógicos programáveis. Nasce assim, uma interface padrão permitindo que pessoas com diferentes habilidades e formações, criem programas durante estágios diferentes do ciclo de vida de um software. Fazem parte deste ciclo: especificação, projeto, implementação, teste, instalação e manutenção.

3.1 IEC 61131- 3 (Linguagens de programação)

A adoção da IEC 61131-3 pelos diversos fabricantes de sistemas de controle é uma realidade inegável. Assim, todo profissional da área, seja técnico ou engenheiro deve conhecê-la.

Hoje, a IEC 61131-3 é o único padrão global para programação de controle industrial que consiste na definição da linguagem que é a ***Função gráfica de seqüenciamento (SFC)***, usada para estruturar a organização interna do programa, e de quatro linguagens, sendo duas textuais: ***Lista de Instrução (IL)*** e ***Texto Estruturado (ST)*** e duas gráficas: ***Diagrama de blocos de funções (FBD)*** e ***Diagrama Ladder (LD)***.

Cabe ao projetista/programador escolher a linguagem que melhor se adapta ao seu sistema, daí a necessidade de conhecer uma pouco de cada uma, não faz parte do escopo deste trabalho detalhar cada uma delas e sim cita-las descrevendo suas características e apresentando um modelo.

3.1.1 Sequential Function Chart (SFC):

São gráficos de função seqüencial, originou-se na França e teve como base a redes de petri e o Grafcet (*Graphe Fonctionnel de Command Etape Trasition*), em 1988 foi publicado tornando-se padrão internacional.

Muito mais que uma linguagem o SFC descreve o comportamento do programa, seja ele, seqüencial paralelo ou misto, além de organizar a sua estrutura interna, ajudando a decompor um problema de controle em partes gerenciáveis, enquanto mantém uma visão global da solução do problema.

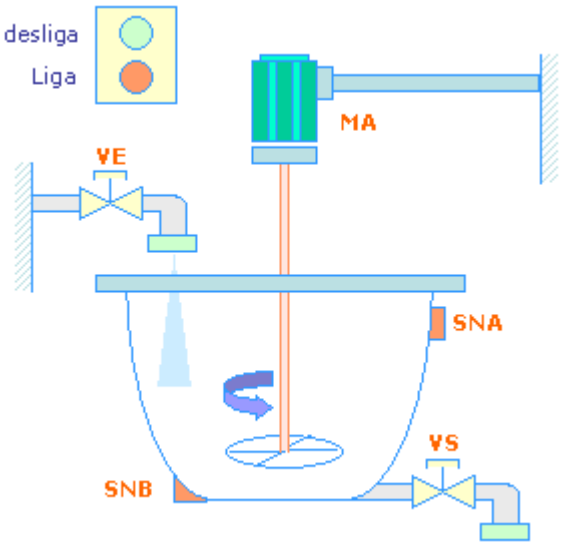
Principais características

- É usada na estruturação do programa, não importando a linguagem utilizada.
- Fácil representação e interpretação
- Facilidade de diagnóstico (localização de falhas)
- Permite gerar divergências e convergências de seqüências.
- Descreve o comportamento do sistema através de passo transições e ações. Sendo:
  - Passo: estado do programa onde as ações são executadas.
  - Transição: condição pela qual o programa muda de estado, passando de um ou mais passos antecessores para um ou mais passos sucessores.
  - Ação: atividade de controle executada num determinado passo.

3.1.1.1 – Implementação prática em SFC - Tanque agitador

Deseja-se implementar um sistema de controle para um tanque misturador simples, como mostrado no esquema:

Ilustração do sistema



Definindo I/O

Entradas

BL = Botão de liga
BD = Botão de desliga
SNA = Sensor nível alto
SNB = Sensor nível baixo

Saídas

VE = válvula de entrada
MA = Motor de Agito
VS = válvula de saída

Figura 3.1 – Tanque agitador

Descrição de funcionamento:

Ao pressionar o botão de liga (BL) a válvula de entrada (VE) é acionada e o tanque começa a encher. Quando o sensor de nível alto (SNA) for atingido, a válvula de entrada (VE) é fechada ligando o motor de agito (MA) que permanece ligado por 10 segundos. Em seguida a válvula de saída (VS) é ligada, quando o sensor de nível baixo (SNB) for acionado o ciclo recomeça. Se o botão de desliga (BD) não for pressionado o ciclo recomeça.

A estrutura do SFC para o tanque agitador é mostrada na figura 3.2.

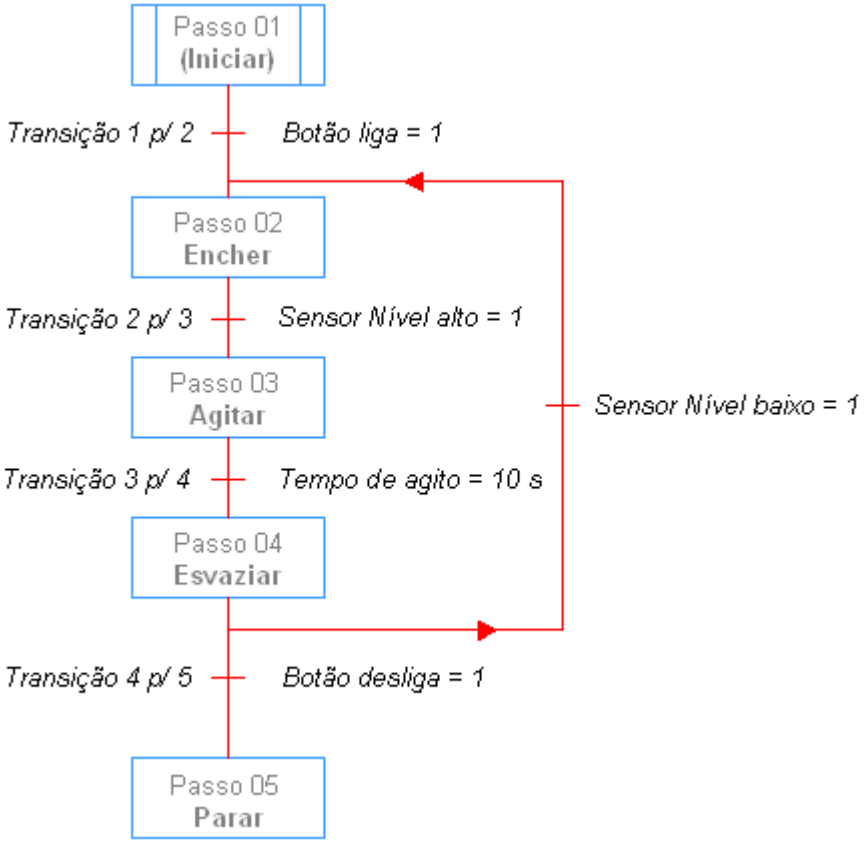


Figura 3.2 - A estrutura do programa SFC

3.1.2 Instruction List (IL)

É uma linguagem textual, próxima do código de máquina, é ideal para resolver problemas simples onde existem poucas quebras no fluxo de execução. Na verdade é apenas uma linguagem adicional, menos amigável e flexível e que deve ser usada para produzir código otimizado para trechos de performance crítica em um programa.

Principais características:

- Linguagem de Baixo Nível
- Semelhante ao *Assembler*
- Ideal para pequenas aplicações ou otimização de códigos
- Linguagem básica para exportação de programas (Portabilidade)

3.1.2.1 Implementação prática em IL – Função ou exclusivo

Deseja-se implementar uma função OU exclusivo, ou seja, fornece 1 (um) à saída quando as variáveis de entrada forem diferentes entre si.

Ilustração do sistema elétrico

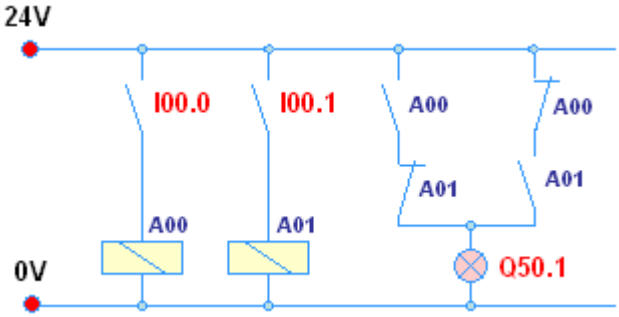


Tabela verdade		
I00.0	I00.1	Q50.0
0	0	0
0	1	1
1	0	1
1	1	0

Figura 3.3 – OU exclusivo elétrico

A lista de instrução para a função OU exclusivo

```
LD      I00.0     * carrega a entrada I00.0
ANDN    I00.1     * faz um and lógico entre I00.0 e I00.1 invertido
OR
(
LDN     I00.0     * carrega a entrada I00.0 invertida
AND     I00.1     * faz um and lógico entre I00.0 invertido e I00.1
)                 * faz o OU lógico entre as duas expressões
ST      Q50       * carrega a saída Q50.0
```

Figura 3.4 – Estrutura do programa IL

3.1.3 Structured Text (ST)

Também é uma linguagem textual, porém de alto nível, que permite a programação estruturada. A vantagem do texto estruturado esta na utilização de sub-rotinas para executar diferentes partes de uma função de controle.

Principais características:

- ◊ Linguagem de alto nível
- ◊ Semelhante ao Pascal (ISO 7185)
- ◊ Ideal para:
  - Tomada de decisões
  - Declarações (Variáveis, POUs, Configurações, etc.)
  - Cálculos
  - Implementação de algoritmos
  - Definição de ações (SFC)
  - Utilização de literais
  - Criação de blocos Etc.

3.1.2.1 Implementação prática em ST – Liga/ desliga motor

O motor (**M**) ficará energizado se, e somente se, o botão liga (**I1**) for acionado e o botão desliga (**I0**) não for acionado. Quando o motor (**M**) estiver energizado, o indicador luminoso (**L**) também estará energizado. (**M**) e (**L**) ficarão desenergizadas caso o botão desliga (**I0**) seja acionado.

O esquema elétrico

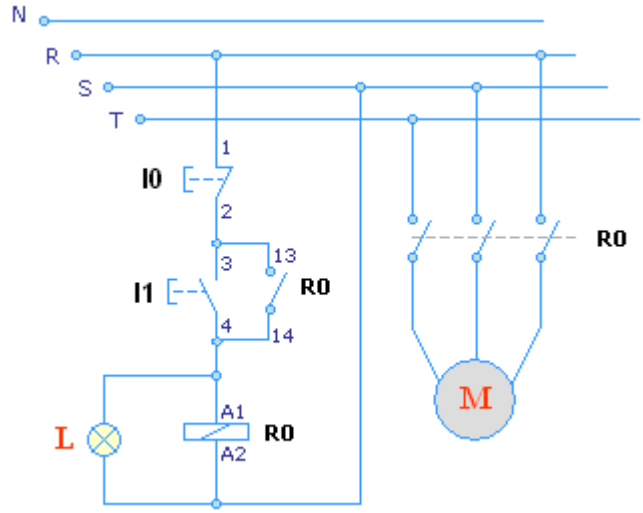


Figura 3.5 – Partida direta de motores

Texto estruturado

```
IF I1
OR M AND N I0
THEN SET M
IF M
THEN SET L
OTHRW RESET M
```

Figura 3.6– Estrutura do programa ST

3.1.4 Function Block Diagram (FBD)

É uma linguagem gráfica, e por isso é muito mais amigável que as textuais é baseada nos circuitos lógicos, portanto muito semelhante as portas lógicas estudadas em eletrônica digital. Também pode ser usado para modelar sistema em termos do fluxo de sinais entre elementos de processamento.

Principais características:

- ◊ Adequada para controle discreto, seqüencial, regulatório, etc.
- ◊ Representação de fácil interpretação
- ◊ Blocos expansíveis em função do no de parâmetros de entrada
- ◊ São disparados por parâmetros externos, enquanto os algoritmos internos permanecem escondidos.
- ◊ Blocos encapsulam o algoritmo, destacando o fluxo de informações e o processamento de sinais.

3.1.4.1 Implementação prática em FBD – segurança em prensas

Duas chaves devem comandar uma prensa simultaneamente de modo que acionada a primeira chave, não podem transcorrer mais do que 0,5s até que a segunda chave seja acionada. Se o operador retirar a mão das chaves, a prensa deverá parar, por razões de segurança.

Ilustração

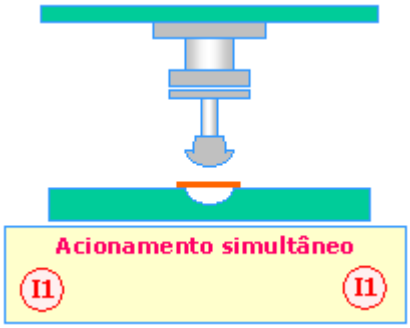


Figura 3.7 – Ilustração da máquina prensa peças

Diagrama de blocos funcionais

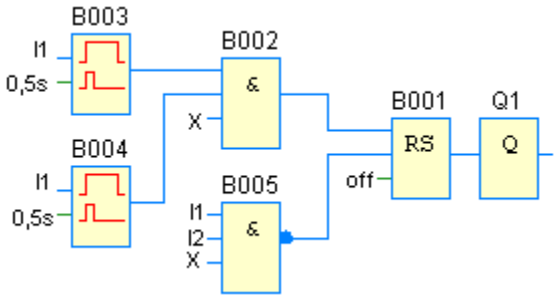


Figura 3.8– Estrutura do programa FBD

3.1.5 Ladder Diagram (LD)

É uma linguagem gráfica, muito amigável, foi baseada na lógica de contatos o que a torna de fácil compreensão no meio elétrico. É a linguagem foco deste trabalho, assim no próximo a linguagem ladder será detalhada.

Principais características:

- Baseada no diagrama elétrico de contatos
- Adequada para controle discreto, combinacional e seqüencial
- Utilizam blocos de função para controle regulatório e funções especiais.

3.1.5.1 Implementação prática em LD – Partida direta reversa

Deseja-se implementar em ladder uma partida direta reversa de motores trifásicos, que consiste em mudar o sentido de rotação de um motor trifásico. Sua seqüência operacional é bastante simples:

Pressionando (S1) energiza-se o contator (K1), fechando o seu selo (13,14) e abrindo o intertravamento (21,22) mesmo pressionando (S2) o contator (K2) não será energizado, devido ao intertravamento, sendo necessário seu desligamento para religar (S2) novamente e a rotação será contrário.

O esquema elétrico da partida direta reversa

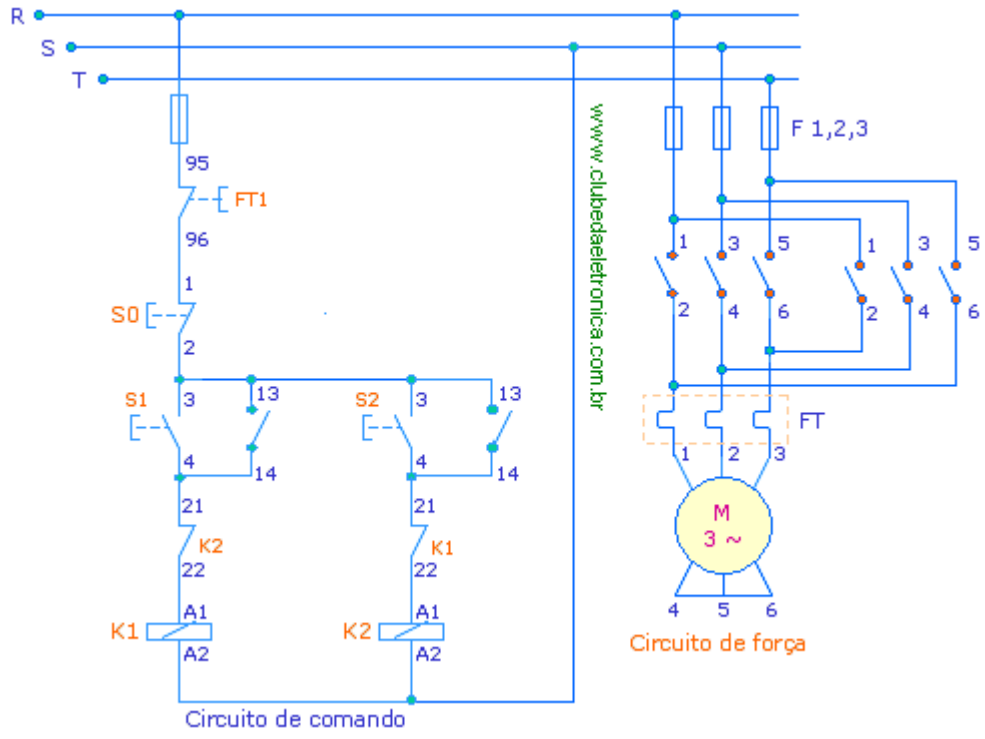


Figura 3.9– Partida direta e reversa

Ladder correspondente

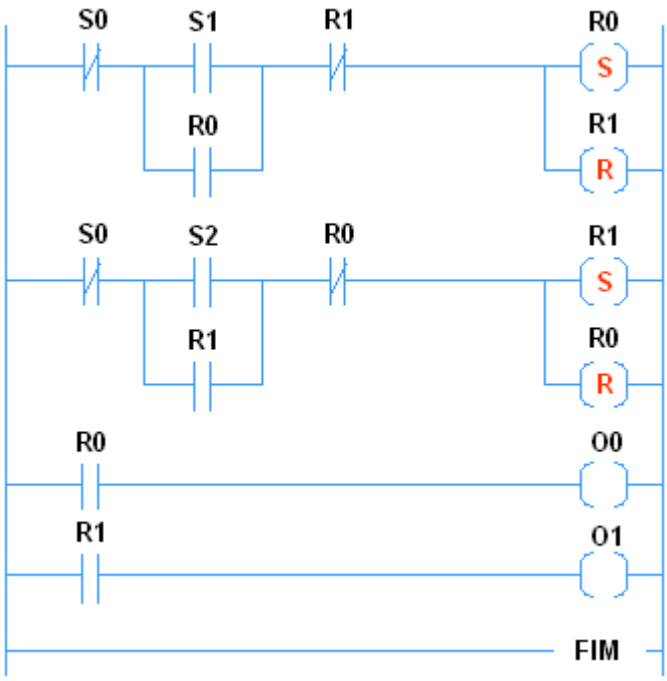


Figura 3.10– Estrutura do programa LD

Dentre as linguagens descritas, daremos um enfoque ao diagrama ladder, a razão é simples é a mais utilizada na indústria. Assim, será trabalhada no próximo capítulo.

Há duas formas de enfrentar dificuldades: alterá-las ou alterar sua maneira de enfrentá-los.

Phullis Bottome.

www.clubedaeletronica.com.br

Referências bibliográficas:

- http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/
- http://www.cpdee.ufmg.br/~carmela/NORMA%20IEC%201131.doc
- http://www.software.rockwell.com/corporate/reference/Iec1131/
- http://www.plcopen.org/
- http://www.lme.usp.br/~fonseca/psi2562%20aula%206%20IHM.pdf
- http://www.teses.usp.br/teses/disponiveis/18/18133/tde-11072002-085859/
- http://www.redenet.edu.br/publicacoes/arquivos/20080108_144615_INDU-058.pdf
- http://www.corradi.junior.nom.br/modCLP.pdf
- http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/

Parte 4 - Técnicas de programação (Lógica simples)

INTRODUÇÃO

Programar em ladder é muito simples, desde que ele tenha uma estrutura sob a qual o programa deve ser desenvolvido, ou seja, se deve ter um modelo de comportamento, obviamente antes de programar em ladder. Este modelo pode ser elaborado de varias maneiras, o importante é ter algo em que se basear um modelo impecável resultará em um programa ladder impecável. As técnicas utilizadas neste trabalho são:

4.1. Lógica combinacional simples: São utilizados em lógica simples sem muitas divergências e convergências, são sugeridos aos que tem familiaridade com sistemas digitais, porém se o modelo ficar muito extenso deve-se minimiza-lo.

4.2. Mapas de Veith-Karnaugh: São utilizados na minimização de sistemas de dificuldade média ou em sistemas onde o comportamento de entradas depende de outras entradas. Se as entradas forem superiores a quatro os mapas não são recomendados.

4.3. Máquina de estados: São utilizados em sistemas de complexos, é de fácil transformação para ladder desde que não haja muitas ramificações.

4.1 LÓGICA COMBINACIONAL SIMPLES

O CLP é um equipamento eletrônico que entre suas aplicações mais simples, esta a execução de funções lógicas em um ambiente industrial. E quando se fala em lógica, logo vêm à mente funções lógicas como “E” ou “AND” e “OU” ou “OR”, muito conhecidas na eletrônica digital. Esta mesma lógica, com algumas mudanças nos símbolos, também pode ser usada na estruturação de programas a serem desenvolvidos em ladder.

Principais blocos

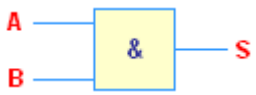
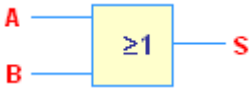
Lógica AND (E)	Expressão lógica	Função executada
	**S = A.B**	Executa função lógica “AND”, ou seja, somente se as entradas A e B estiverem em nível alto a saída S será acionada.

Tabela verdade

A	B	S
0	0	0
0	1	0
1	0	0
1	**1**	**1**

Programa Ladder correspondente

Lógica OR (ou) | **Expressão lógica** | **Função executada**



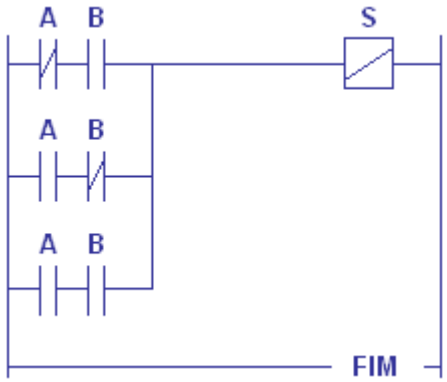
S = A+B

Executa função lógica "OR", ou seja, para que a saída S seja acionada basta que uma das entradas A ou B esteja em nível alto.

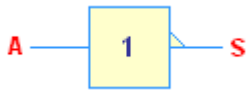
Tabela verdade

A	B	S
0	0	0
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Programa Ladder correspondente



Lógica NOT (não) | **Expressão lógica** | **Função executada**



S = A'

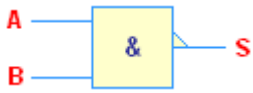
Executa função lógica "NOT", ou seja, nega ou inverte o sinal de entrada.

Tabela verdade

A	S
0	**1**
1	0

Programa Ladder correspondente

Lógica NAND (não e) | **Expressão lógica** | **Função executada**



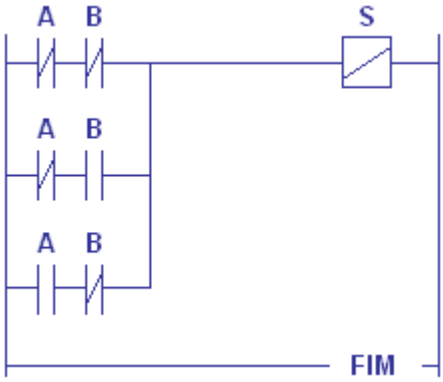
S = (A.B)'

Executa função lógica "NAND", ou seja, nega ou inverte as saídas da função AND.

Tabela verdade

A	B	S
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	1	0

Programa Ladder correspondente



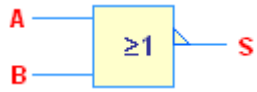
Lógica NOR (não ou)	Expressão lógica	Função executada
	S = (A+B)'	Executa função lógica "NOR", ou seja, nega a função OR, invertendo assim, suas saídas.

Tabela verdade

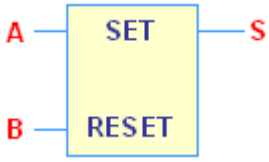
A	B	S
0	**0**	**1**
0	1	0
1	0	0
1	1	0

Programa Ladder correspondente

Blocos de memorização.

São utilizados com muita freqüência, pois tem a função de memorizar um BIT.

SET RESET



Função executada

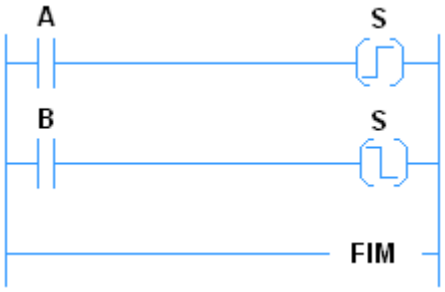
"Set" significa Ligar e "Reset" desligar. Seu funcionamento é simples uma vez setado (nível lógico (1) em A) ele comuta a saída S, ou seja, vai para (1) e somente volta para nível baixo (0) se for resetado.

Tabela verdade

A	B	S
0	**0**	**CA**
0	**1**	**0**
1	**0**	**1**
1	**1**	**CP**

CA= Condição Anterior
CP= Condição Proibida

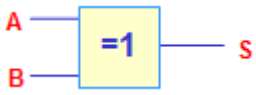
Programa Ladder correspondente



Exercícios com lógica simples

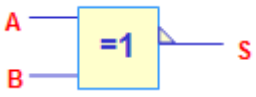
1- Implemente uma lógica XOR (OU exclusivo) em ladder.

I0	I1	O0

Símbolo (FDB)	Ladder correspondente
	

2- Implemente uma lógica XNOR (OU coincidência) em ladder.

I0	I1	O0

Símbolo (FDB)	Ladder correspondente
	

3- Extraia a expressão lógica, monte o circuito lógico (utilize blocos lógicos funcionais) e construa a lógica ladder a partir da tabela verdade.

a)

I0	I1	O0
0	0	0
0	1	1
1	0	0
1	1	1

Ladder aqui e demais respostas (no verso)

b)

I0	I1	O0
0	0	0
0	1	1
1	0	0
1	1	0

Ladder aqui e demais respostas (no verso)

c)

I0	I1	O0
0	0	1
0	1	1
1	0	0
1	1	1

Ladder aqui e demais respostas (no verso)

d)

I0	I1	I3	O0
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Ladder aqui e demais respostas (no verso)

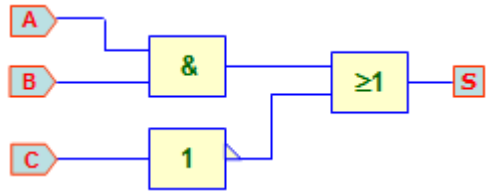
e)

I0	I1	I3	O0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

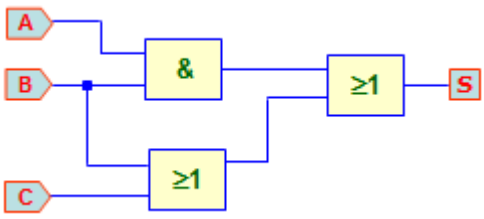
Ladder aqui e demais respostas (no verso)

Converta os diagramas (dado em FBD) para ladder.

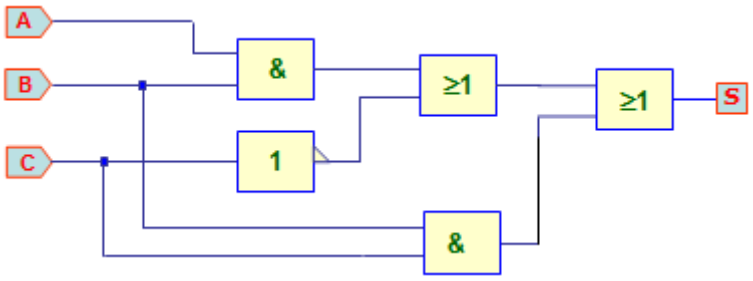
a) Lógica FBD **Ladder correspondente**



b) Lógica FBD **Ladder correspondente**



c) Lógica FBD **Ladder correspondente**



Ladder correspondente

Dado as seguintes expressões lógicas, construa o diagrama correspondente em linguagem ladder e em diagramas de blocos funcionais (FDB).

a) S= (A+B).C

b) S= (A.B)+(C.D)

c) S= (A'+B).(C.D)'

d) S= (A+B).D'

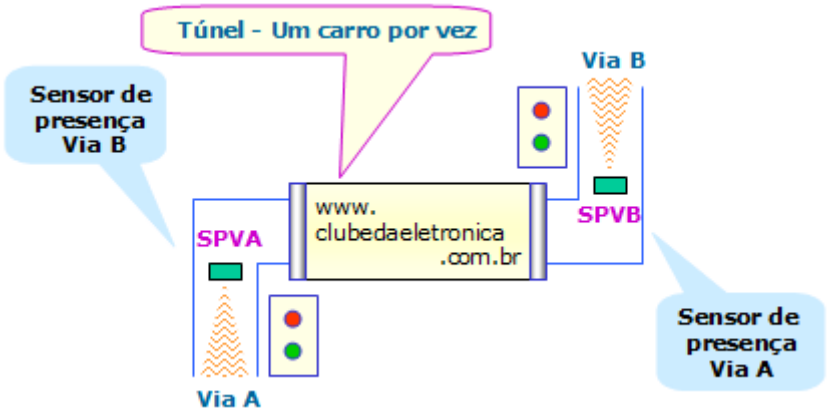
Aplicação da lógica combinacional (Simples)

A aplicação da lógica combinacional é sem dúvidas o que mais interessa nos sistemas digitais, pois pode ser usada em diversas áreas.

Aplicação 01 – Controle de trafego (resolvido)

Deseja-se programar um controle de trafego para um túnel que só permite a passagem de um carro por vez.

Veja ilustração:



A prefeitura que encomendou o projeto tem os seguintes critérios:

Quando os sensores detectarem a presença do carro, um nível lógico alto (ON) será enviado ao seu respectivo dispositivo de atuação.

Situação dos sensores		Critérios de projeto
SPVA (SA)	**SPVB (SB)**	
OFF	**OFF**	Se não houver nenhum carro, a **via B deverá ser liberada (verde)** e a **via A bloqueada (vermelho)**.
OFF	**ON**	**Se o sensor detectar carro na** via B, esta será liberada (sinal verde) **e a** Via A bloqueada (sinal vermelho)**.**
ON	**OFF**	Se o sensor detectar carro na **via A, esta será liberada (sinal verde)** e a **Via B bloqueada (sinal vermelho)**.
ON	**ON**	Se ambos os sensores detectarem carros, a via A deverá ser liberada (sinal verde) e a **via B bloqueada (sinal vermelho)**.

1º Passo – Montar a tabela verdade a partir de todas as condições possíveis

SPVA (SA)	SPVB (SB)
0	0
0	1
1	0
1	1

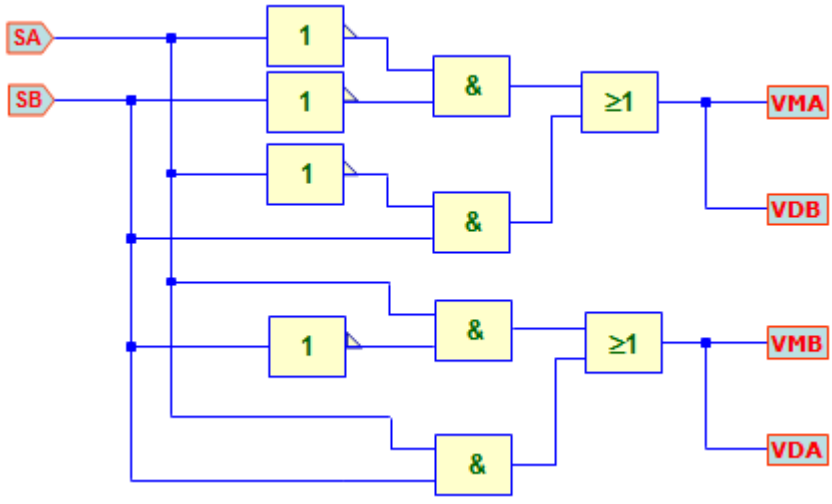
VMA	VDA	VMB	VDB
1	0	0	1
1	0	0	1
0	1	1	0
0	1	1	0

2º Passo – Extrair a tabela verdade das expressões verdadeiras

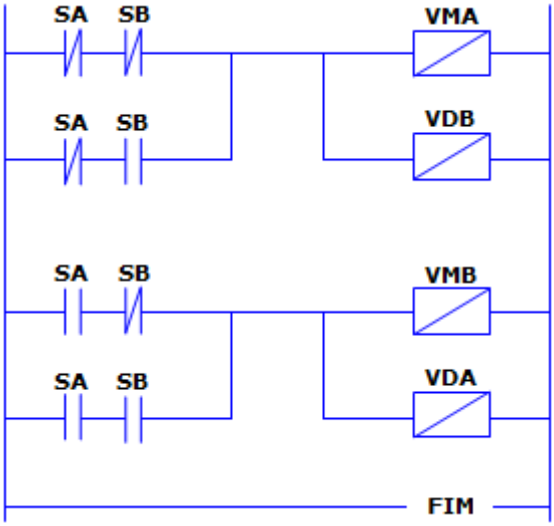
$$VMA = VDB = (SA' . SB') + (SA' . SB)$$

$$VMB = VDA = (SA . SB') + (SA . SB)$$

3º Passo – Montar o circuito lógico



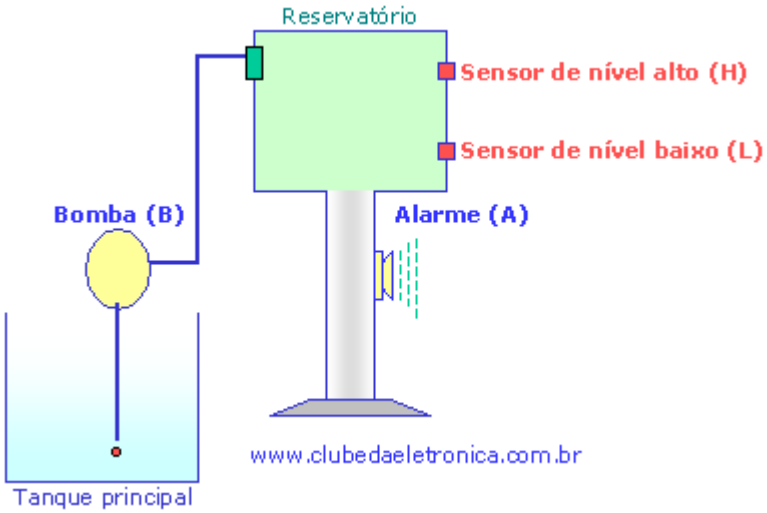
4º Passo – Montar o programa ladder



Praticando...

1- Aplicação 2 – controle de nível

Deseja-se controlar o nível de água de um reservatório, conforme ilustração:

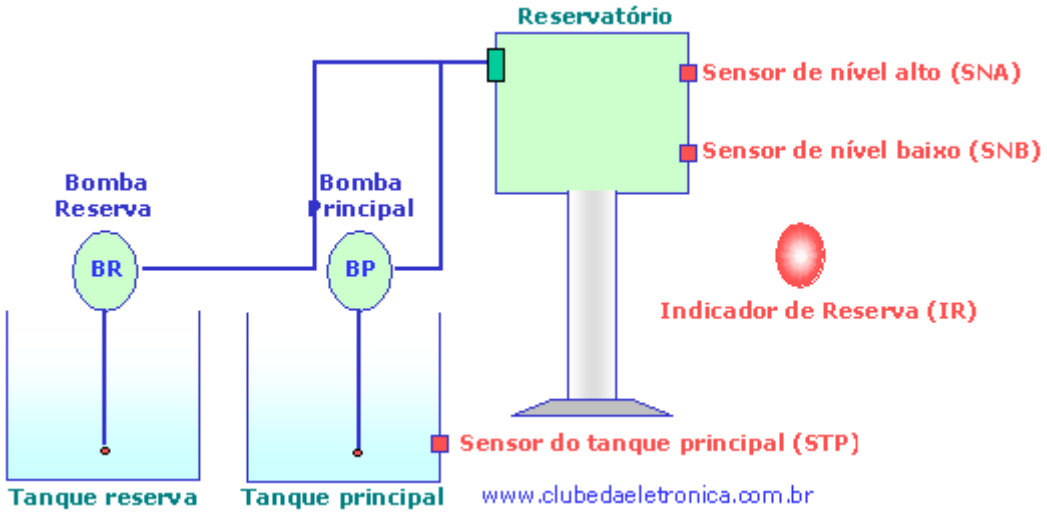


Descrição de funcionamento:

- O reservatório deve estar sempre cheio, ou seja, H=1;
- Se H=0, a bomba deverá ser acionada;
- Se a bomba não atender a demanda e o reservatório esvaziar, ou seja, L=0, um alarme deverá ser acionado.

2- Aplicação 3 – controle de nível com tanque reserva

Deseja-se controlar o nível de água de um reservatório, conforme ilustração:



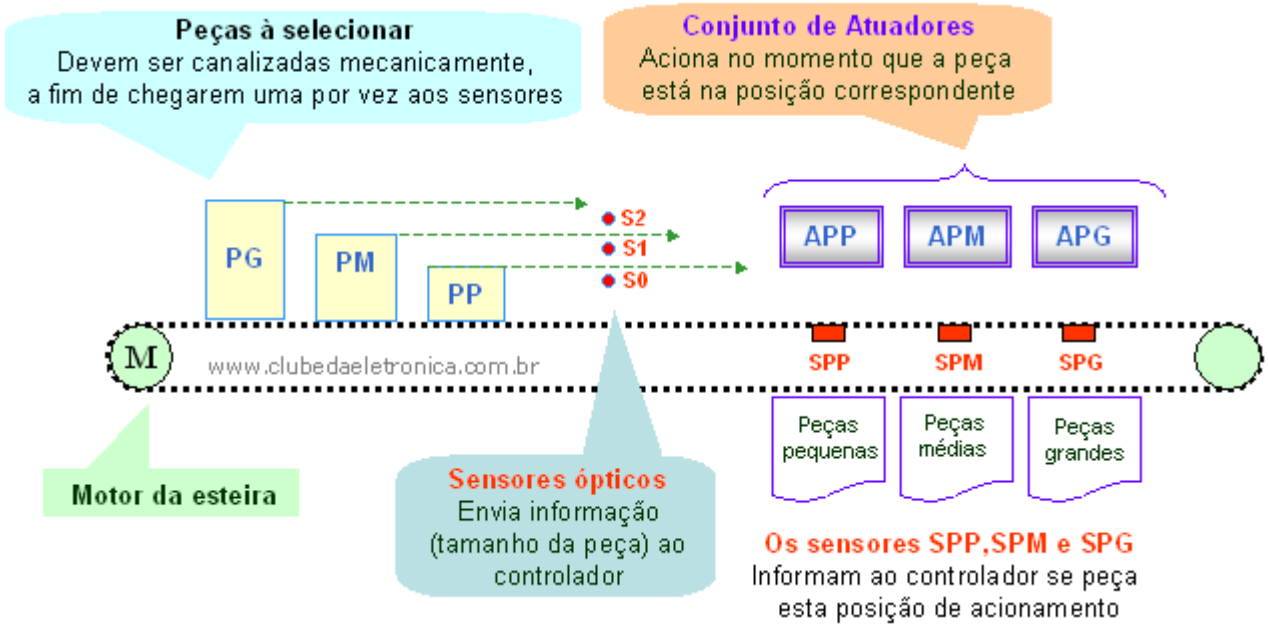
Seu funcionamento deve ser o seguinte:

- O reservatório deve estar sempre cheio, ou seja, SNA=1;
- Se SNA=0, a bomba principal BP deverá ser acionada, mas somente se houver água no tanque principal, ou seja, STP =1, se STP =0, a bomba reserva deve ser acionada;
- Se a bomba reserva BR for acionada, um indicador de reserva (IR) deverá ser acionado.

3- Aplicação 4 – Selecionar de peças (Resolvido)

Deseja-se implementar um selecionador de peças pequenas, médias e grandes. O sistema consiste dois sensores S1 e S2 que selecionarão as peças e três atuadores sendo um para cada tipo de peça que deverão colocar cada peça em seu respectivo compartimento.

Ilustração simplificada:



Descrição:

- Se nenhum sensor for ativado, então a peça é pequena.
- Se somente o sensor S1 for ativado, então a peça é média.
- Se os dois sensores forem ativados então a peça é grande.

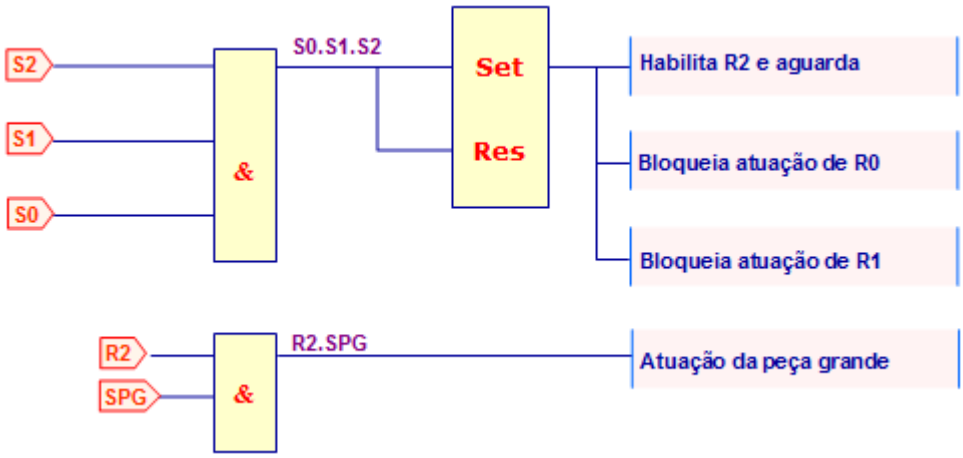
Nota: O processo é continuo e somente haverá uma atuação por vez.
Haverá um alimentador (não incluso) que colocará uma peça por vez com intervalo de tempo pré-definido entre elas.

Descrição das etapas:

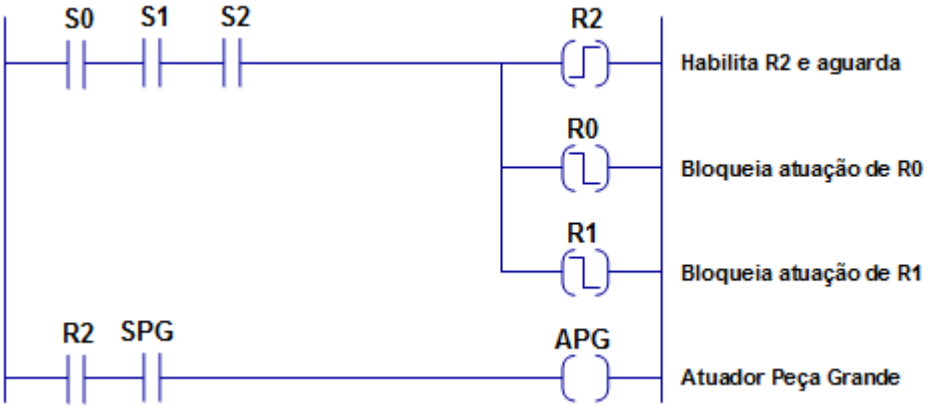
Peça grande ⇒ Se S1, S2 e S3 forem cortados, "setará" um contato auxiliar R0 que fica aguardando a posição atuador de peça grande (SPG) e se esta for alcançada a peça será retirada.

A peça grande só será retirada se as peças pequenas ou médias não estiverem aguardando o sensor de suas posições.

O diagrama de blocos para peça grande



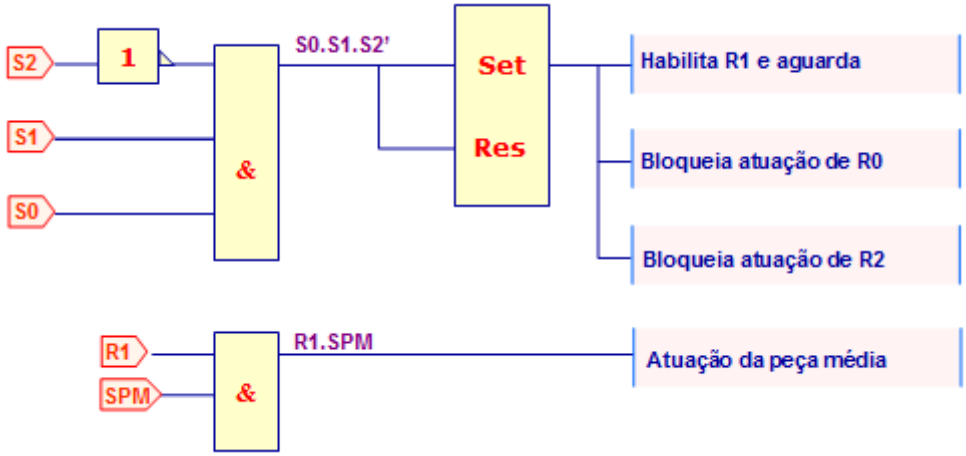
Ladder correspondente para peça grande



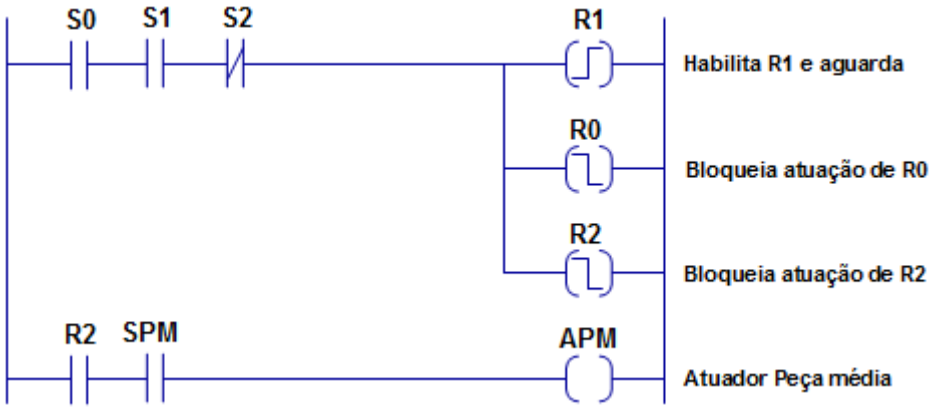
Peça média ⇒ Se S0 e S1 forem cortados e S2 não "setará" um contato auxiliar R1 que fica aguardando a posição atuador de peça média (SPM) se esta for alcançada a peça será retirada.

A peça média só será retirada se as peças pequenas ou grandes não estiverem aguardando o sensor de suas posições.

O diagrama de blocos para peça média



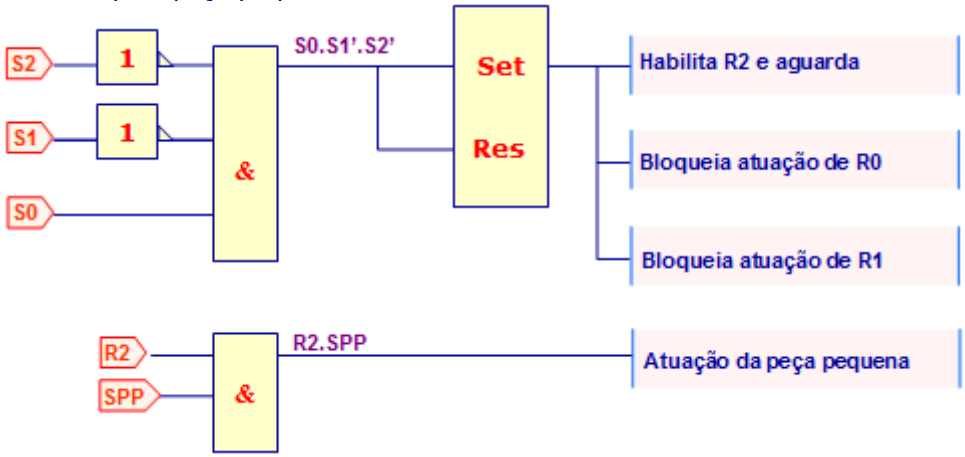
Ladder correspondente para peça média



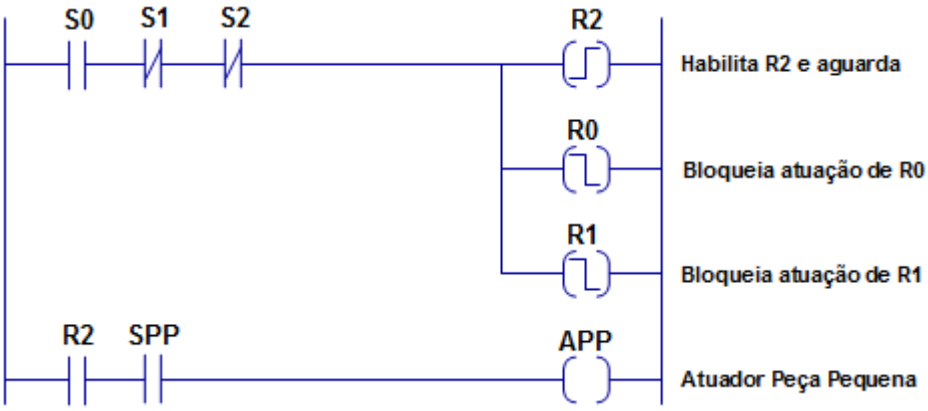
Peça pequena ⇒ Se somente S0 for cortado, "setará" um contato auxiliar R2 que fica aguardando a posição atuador de peça pequena (SPP) e se esta for alcançada a peça será retirada.

A peça pequena só será retirada se as peças médias ou grandes não estiverem aguardando o sensor de suas posições.

O diagrama de blocos para peça pequena

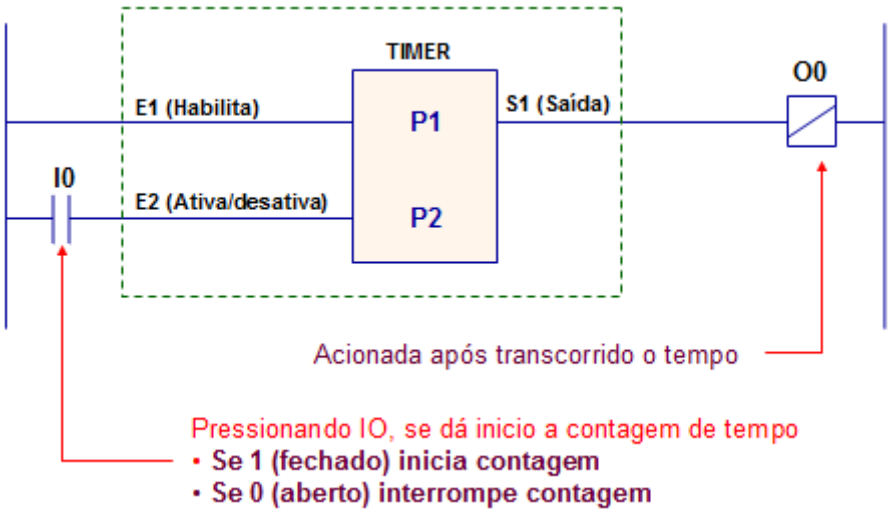


Ladder correspondente para peça pequena



Bloco timer (temporizador)

Este elemento, como o próprio nome diz, tem a finalidade de contar o tempo. Uma vez carregado um determinado período de tempo como parâmetro e tendo a contagem sido habilitada, este valor é decrementado de 10 ms até que chegue a zero, momento em que a saída do bloco é ativada indicando o fim da contagem.



Parâmetros:

P1 – Representa o valor corrente da contagem do temporizador e deve ser obrigatoriamente uma memória inteira (operador M).

P2 - Representa o valor inicial da contagem e deve ser obrigatoriamente uma memória inteira (operador M) ou uma constante inteira (operador K)

Entradas:

E1 – Energizada habilita o bloco temporizador, permitindo a contagem de tempo (se ativado). Se desenergizado o temporizador será desativado.

E2 – Se energizada (1) ativa a contagem do tempo e desenergizada (0) o temporizador fica em seu estado de reset, ou seja, não conta.

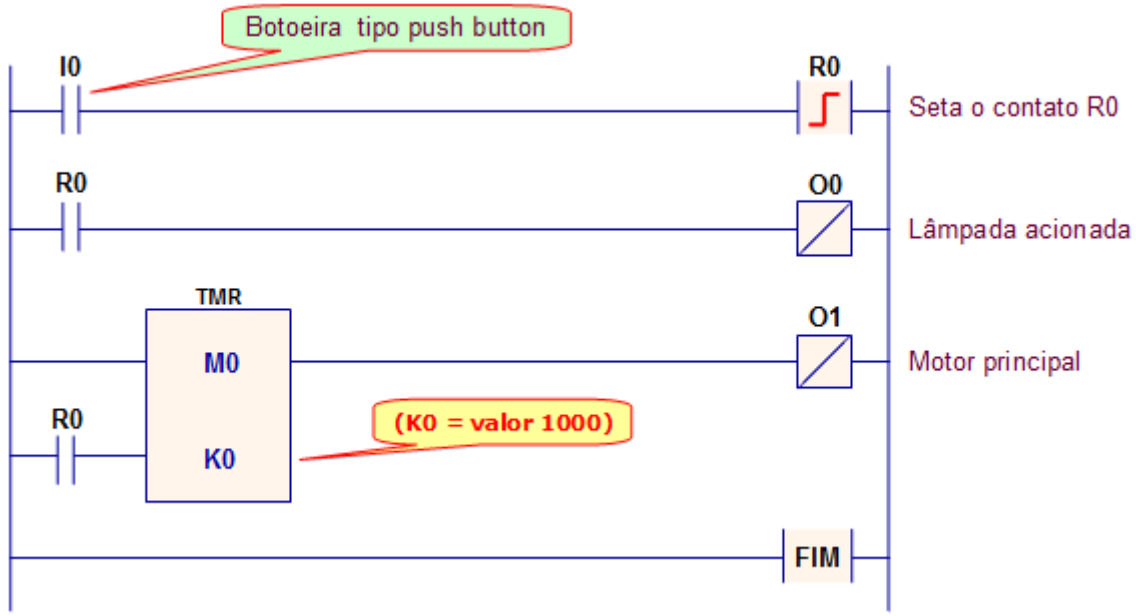
Saída:

S1 – Se ativa (1) indica que o tempo expirou, ou seja, fim da contagem. Se inativa indica que o tempo ainda não terminou ou que o temporizador está desabilitado.

Aplicações práticas do temporizador

1- **(Resolvido)** Deseja-se implementar um sistema em que ao pressionar BL (push button), acenderá imediatamente uma lâmpada indicando que a máquina esta energizada e após 10s ligará o motor principal.

Solução:



2 - Implemente no mesmo programa um botão de desliga, também do tipo push butom.

4- Elabore um programa que ao pressionar BL (push button) ligará uma lâmpada instantaneamente e somente desligará 5 segundos após BD (push button) ter sido pressionado.

5- Elabore um programa que ao pressionar BL ligará 4 motores em seqüência, sendo o primeiro instantaneamente e os demais respeitando um intervalo de 4 segundos. Pressionando BD, todos pararão imediatamente.

6- Construa um programa capaz de energizar 5 motores em sequência M1, M2, M3, M4 e M5 quando BL for pressionado e que desligue também na mesma seqüência, ou seja, M1, M2, M3, M4 e M5. Utilize um intervalo de tempo de 6s.

"Se você fica esperando, tudo o que acontece é que você fica velho."

(Larry McMurtry)

www.clubedaeletronica.com.br

Referências bibliográficas:

- http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/
- http://www.cpdee.ufmg.br/~carmela/NORMA%20IEC%201131.doc
- http://www.software.rockwell.com/corporate/reference/Iec1131/
- http://www.plcopen.org/
- http://www.lme.usp.br/~fonseca/psi2562%20aula%206%20IHM.pdf
- http://www.teses.usp.br/teses/disponiveis/18/18133/tde-11072002-085859/
- http://www.redenet.edu.br/publicacoes/arquivos/20080108_144615_INDU-058.pdf
- http://www.corradi.junior.nom.br/modCLP.pdf
- http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/

Parte 05 - Técnicas de programação (mapas de Veitch-Karnaugh)

Mapas de Veitch-Karnaugh

Montar circuitos lógicos a partir de tabela verdade, embora seja tarefa fácil, geral um circuito extremamente grande. A fim de minimiza-los foram criados os mapas de Veitch-Karnaugh, o nome deve-se aos seus criadores Edward Veitch e Maurice Karnaugh.

Um mapa de Karnaugh é uma ajuda excelente para simplificação de funções de até 4 variáveis. Para funções de mais de 4 variáveis a simplificação é mais complexa pois torna-se uma tarefa árdua identificar as células adjacentes no mapa. Para funções de mais de 4 variáveis devem ser utilizadas soluções algorítmicas computacionais.

O método utiliza como base uma tabela verdade onde serão colocadas todas os variáveis de entrada e saídas.

Tabela verdade **Posição no mapa**

A	B	C	D	S
0	0	0	0	1ª Linha - 1ª coluna
0	0	0	1	1ª Linha - 2ª coluna
0	0	1	0	1ª Linha - 4ª coluna
0	0	1	1	1ª Linha - 3ª coluna
0	1	0	0	2ª Linha - 1ª coluna
0	1	0	1	2ª Linha - 2ª coluna
0	1	1	0	2ª Linha - 4ª coluna
0	1	1	1	2ª Linha - 3ª coluna
1	0	0	0	3ª Linha - 1ª coluna
1	0	0	1	3ª Linha - 2ª coluna
1	0	1	0	3ª Linha - 4ª coluna
1	0	1	1	3ª Linha - 3ª coluna
1	1	0	0	4ª Linha - 1ª coluna
1	1	0	1	4ª Linha - 2ª coluna
1	1	1	0	4ª Linha - 4ª coluna
1	1	1	1	4ª Linha - 3ª coluna

Mapa de Karnaugh (4 variáveis)

Saída: C D / A B	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

Nota:
Os valores são ordenados seguindo o código Gray, ou seja, só varia um bit por vez.

Entendendo os mapas através de exemplos:

Duas variáveis

Tabela verdade

A	B	S
0	0	0
0	1	0
1	0	1
1	1	1

2ª linha A=1
(Não muda e, portanto, considera)

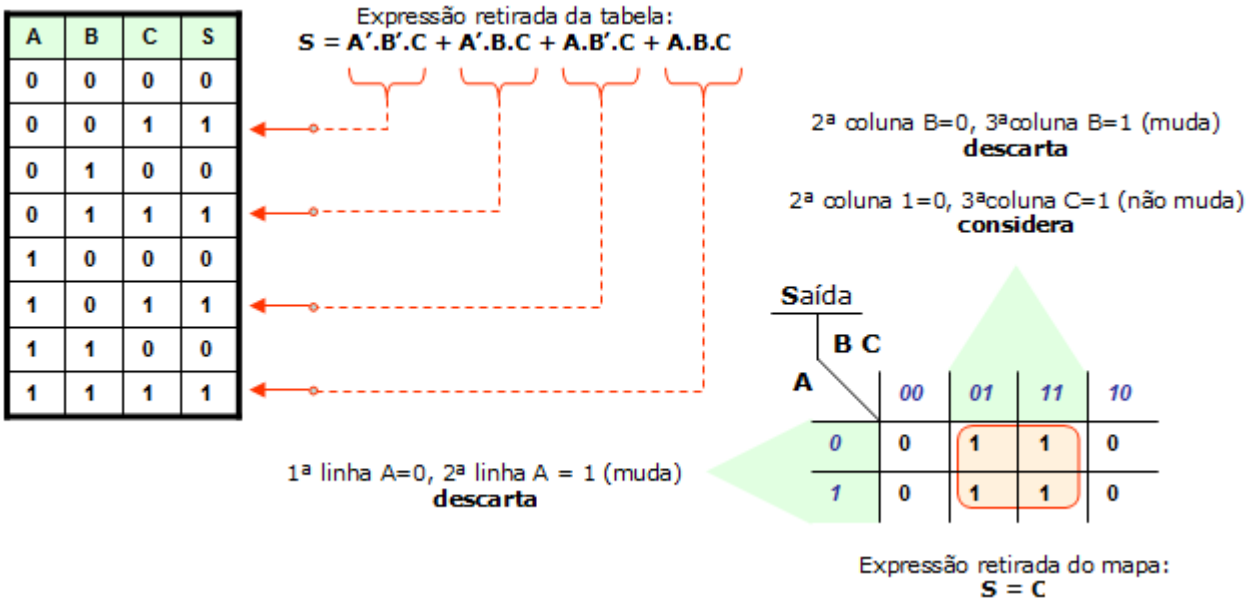
1ª coluna B=0, 2ªcoluna B=1
(Muda e, portanto, descarta)

Saída: B / A	0	1
0	0	0
1	1	1

Expressão retirada da tabela:
S = A.B' + A.B

Expressão retirada do mapa:
S = A

Três variáveis



Praticando...

1- Dados os mapas extraia as expressões e monte o ladder.

S (AB \ CD)	00	01	11	10
00	0	0	0	1
01	1	1	1	1
11	0	0	0	1
10	1	1	1	1

S=

S (AB \ CD)	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	0	0	1
10	1	0	0	1

S=

S (AB \ CD)	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	0	0	1
10	1	0	0	1

S=

S (AB \ CD)	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

S=

S (AB \ CD)	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	0	0	1
10	1	1	1	1

S=

S (AB \ CD)	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

S=

S (AB \ CD)	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	0	0

S=

S (AB \ CD)	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	0	0	1
10	1	1	1	1

S=

S (AB \ CD)	00	01	11	10
00	1	1	1	1
01	0	1	1	0
11	0	1	1	0
10	1	1	1	1

S=

2- Dado a tabela verdade extraia o mapa e a expressão lógica e o ladder.

Tabela 1				
A	B	C	D	S
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Tabela 2				
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabela 3				
A	B	C	D	S
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

S / AB \ CD	00	01	11	10
00				
01				
11				
10				

S=

S / AB \ CD	00	01	11	10
00				
01				
11				
10				

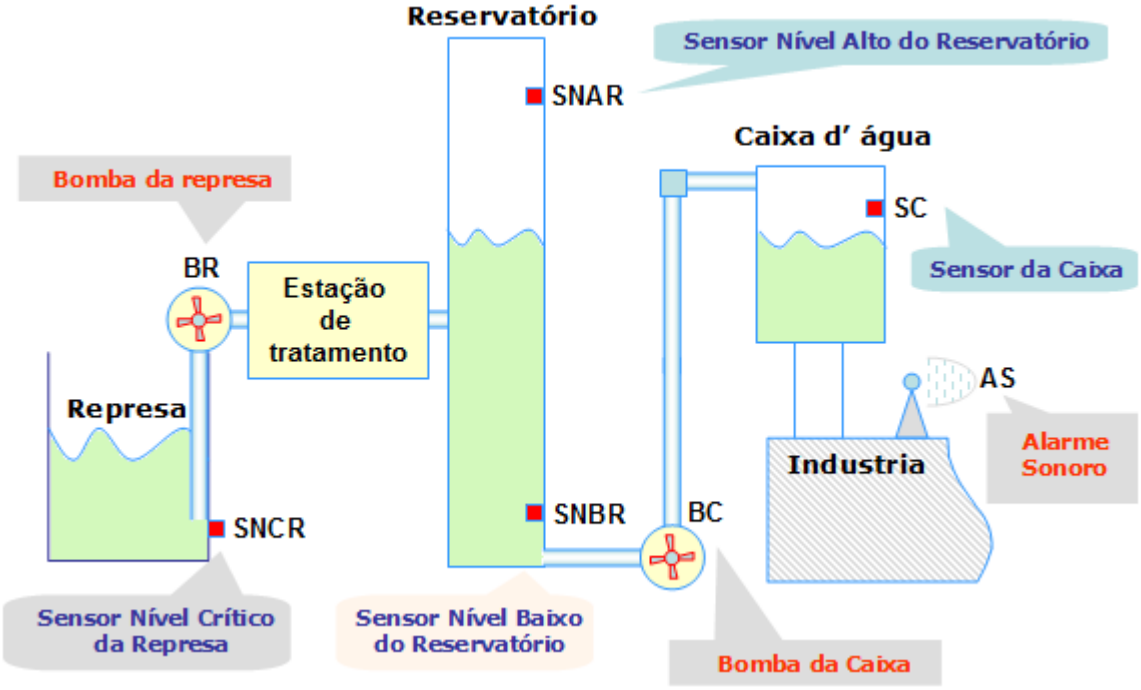
S=

S / AB \ CD	00	01	11	10
00				
01				
11				
10				

S=

Aplicação sistema de abastecimento de água (resolvido)

Uma indústria capta toda água que precisa de uma represa local. Esta água é bombeada para uma estação de tratamento e em seguida armazenada em um reservatório e esta por sua vez deve ser bombeada à uma caixa de água de menor porte, a fim de alimentar a industria.



Descrição do processo

Sempre que o sensor de nível alto do reservatório **(SNAR)** estiver desacionado (0), a bomba do rio **(BR)** deve ser ligada (1) para encher o reservatório até o sensor de nível alto **(SNAR)** ser acionado (1).

A indústria esta em uma região de baixo índice pluviométrico e o rio, as vezes, fica baixo não sendo possível captar a água. Então o sensor de nível crítico do rio **(SNCR)** estiver desacionado (0), um alarme **(AS)** deverá ser ligado (1) para avisar o operador e a bomba do rio **(BR)** deve ser desligada (0).

Ao mesmo tempo a caixa d'água da indústria deve ficar com seu nível sobre o sensor da caixa (SC), ou seja, **SC** = 1. Se o nível da caixa d'água ficar abaixo de **SC**, ou seja, **SC** = 0 a bomba da caixa **(BC)** deve ser ligada (1), mas somente se **SNBR** = 1.

Se ocorrer um erro lógico todas as saídas deverão ser desligadas e um indicador de **ERRO** acionado

Definindo Entradas e Saídas

Entradas

Nomes	Siglas
Sensor de nível alto do reservatório	**SNAR**
Sensor de nível baixo reservatório	**SNBR**
Sensor de nível crítico do rio	**SNCR**
Sensor da caixa	**SC**

Saídas

Nomes	Siglas
Bomba do rio	**BR**
Bomba da caixa	**BC**
Alarme	**AL**
Erro	**ERRO**

Tabela verdade

A tabela verdade é uma tabela onde são pressupostas todas as condições possíveis de entrada do sistema, e de acordo com estas entradas a saída poderá ser verdadeira ou não.

O número de condições possíveis das entradas depende do número de entradas, pode ser calculado pela seguinte expressão:

$$\text{condições possíveis} = 2^{n^o \text{ de entradas}}$$

Logo, condições possíveis = 2^4 = 16.

Para saber se a saída será verdadeira (1) ou falsa (0), devemos analisas as 16 condições. Dentre estas 16 condições algumas nunca acontecerão e serão consideradas irrelevantes e a saída será representada por um x.

Tabela verdade

	Entradas			
	SNAR	SNBR	SNCR	SC
00	0	0	0	0
01	0	0	0	1
02	0	0	1	0
03	0	0	1	1
04	0	1	0	0
05	0	1	0	1
06	0	1	1	0
07	0	1	1	1
08	1	0	0	0
09	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Saídas			
BR	BC	AL	Erro
0	0	1	0
0	0	1	0
1	0	0	0
1	0	0	0
0	1	1	0
0	0	1	0
1	1	0	0
1	0	0	0
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
0	1	1	0
0	0	1	0
0	1	0	0
0	0	0	0

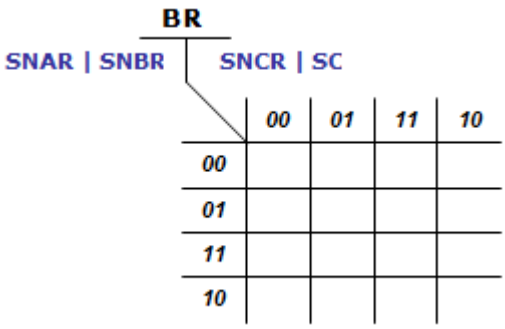
Os mapas de Veith-Karnaugh

Uma vez montada a tabela verdade, poderíamos até extrair a expressão booleana para cada saída, porém ela fica muito grande o que inviabiliza sua criação lógica.

Os mapas de Veith-karnaugh minimizam estas expressões e assim, facilitam a criação do circuito lógico.

Elaboração do mapa

Todas as variáveis de entradas são combinadas em 16 formas diferentes, então o mapa de Veith-Karnaugh terá 16 posições. O arranjo mais conveniente é em uma matriz 4x4.



Os bits no mapa representam todas as combinações possíveis de entrada (**SNAR, SNBR, SNCR e SC**) ordenados segundo o código de Gray de forma que apenas uma variável muda de valor entre cada célula e uma adjacente.

Os espaços não preenchidos do mapa representam as saídas, que podem ser verdadeiras (1), falsas (0) ou irrelevantes (X).

Mapa da bomba do Rio (BR)

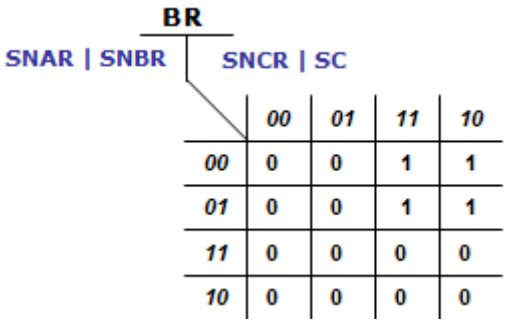
Na tabela verdade as condições em que bomba do rio é verdadeira, ou seja, estará ligada, são:

	Entradas			
	SNAR	SNBR	SNCR	SC
03	0	0	1	0
04	0	0	1	1
07	0	1	1	0
08	0	1	1	1

Saídas			
BR	BC	AL	Erro
1	0	0	0
1	0	0	0
1	1	0	0
1	0	0	0

As demais são falsas, ou seja, estão desligadas (0) ou irrelevantes.

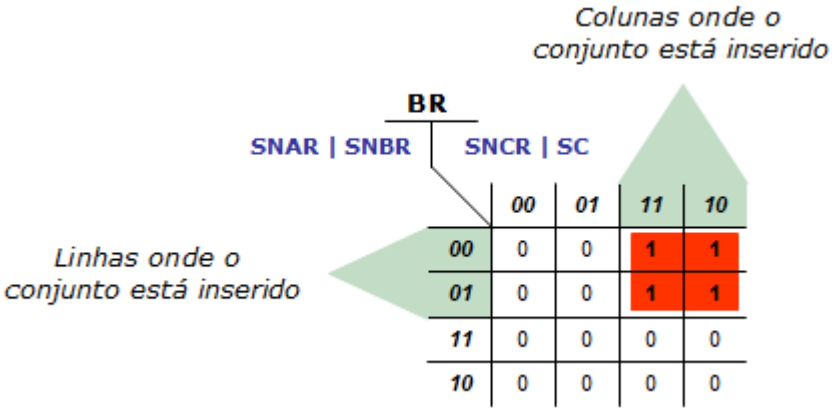
Mapa referente a bomba do rio (BR)



	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	0	0

Após o mapa ter sido construído a próxima tarefa é encontrar os termos mínimos a usar na expressão final. Estes termos são encontrados agrupando conjuntos de saídas verdadeiras (1) adjacentes no mapa. O agrupamento deve ser retangular e deve ter uma área igual a uma potência de 2 (i.e. 2, 4, 8, …). Os retângulos devem ser os maiores possíveis, sem conter nenhum 0.

O agrupamento referente à bomba do rio (BR)



	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	0	0

Agora devemos extrair a expressão booleana correspondente a saída verdadeira (1) para a bomba do rio (BR).

A expressão Booleana referente à saída (BR)

SNAR ⇒ Analisando as linhas onde o conjunto esta inserido, note que a entrada SNAR **não muda**, ou seja, na primeira linha é falsa (0) e na segunda continua falsa (0). Neste caso, a saída somente será verdadeira (1), para SNAR, se ele for invertido.

SNBR ⇒ Ainda analisando as linhas, onde o conjunto esta inserido, note que a entrada SNBR **muda**, ou seja, na primeira linha é falsa (0) e na segunda linha é verdadeira (1). Neste caso, ela deve ser ignorada.

SNCR ⇒ Analisando as colunas, onde o conjunto esta inserido, note que a entrada SNCR **não muda**, ou seja, na primeira coluna é verdadeira (1) e na segunda continua verdadeira (1). Neste caso, a saída será verdadeira (1) sempre que SNCR for verdadeiro.

SC ⇒ Ainda analisando as colunas, onde o conjunto esta inserido, note que a entrada SC **muda**, ou seja, na primeira coluna é verdadeira (1) e na segunda é falsa (0). Neste caso, ela deve ser ignorada.

Cada conjunto representa uma função lógica "AND", assim, a saída **BR** só estará ligada se SNAR for falso (0) e SNCR for verdadeiro. Logo a expressão é:

BR = SNAR' . SNCR

Ladder correspondente à bomba do rio (BR)

Mapa da bomba da caixa (BC).

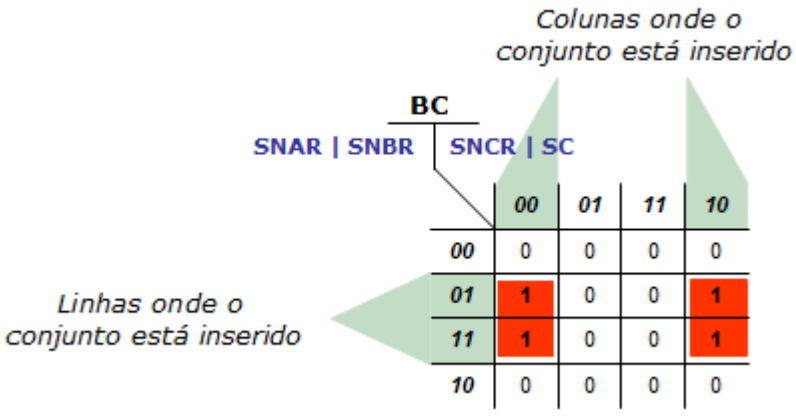
Na tabela verdade as condições em que bomba do rio é verdadeira, ou seja, estará ligada, são:

	Entradas			
	SNAR	SNBR	SNCR	SC
05	0	1	0	0
07	0	1	1	0
13	1	1	0	0
15	1	1	1	0

Saídas			
BR	BC	AL	Erro
0	1	1	0
1	1	0	0
0	1	1	0
0	1	0	0

As demais são falsas, ou seja, estão desligadas.

Mapa referente à bomba da caixa (BC) (Já agrupado)



Agora devemos extrair a expressão booleana correspondente a saída verdadeira (1) para a bomba da caixa (BC).

A expressão Booleana referente à saída *(BC)*

SNAR ⇒ Analisando as linhas onde o conjunto esta inserido, note que a entrada SNAR **muda**, ou seja, na primeira linha é falsa (0) e na segunda é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

SNBR ⇒ Ainda analisando as linhas, onde o conjunto esta inserido, note que a entrada SNBR **não muda**, ou seja, na primeira linha é verdadeira (1) e na segunda linha continua verdadeira (1). Neste caso, a saída será verdadeira (1) sempre que SNBR for verdadeiro.

SNCR ⇒ Analisando as colunas onde o conjunto esta inserido, note que a entrada SNCR **muda**, ou seja, na primeira coluna é falsa (0) e na segunda é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

SC ⇒ Analisando as colunas onde o conjunto esta inserido, note que a entrada SC **não muda**, ou seja, na coluna linha é falsa (0) e na segunda continua falsa (0). Neste caso a saída somente será verdadeira (1) para SC, se ele for invertido.

Cada conjunto representa uma função lógica "AND", assim, a saída BC só estará ligada se SNBR for verdadeiro (1) e SC for falso (0). Logo a expressão é:

BC = SNBR . SC'

Ladder correspondente à bomba da caixa (BC)

Mapa do alarme (AL).

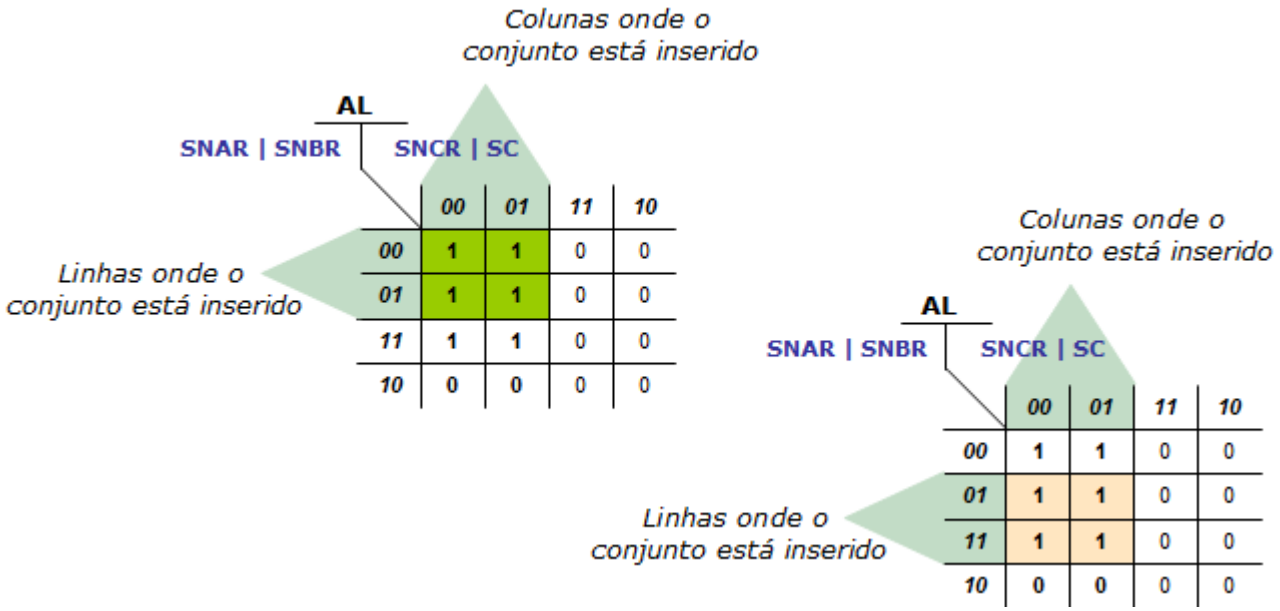
Na tabela verdade as condições em que o alarme é verdadeira, ou seja, onde estará ligado, são:

	Entradas			
	SNAR	**SNBR**	**SNCR**	**SC**
01	**0**	**0**	**0**	**0**
02	**0**	**0**	**0**	**1**
05	**0**	**1**	**0**	**0**
06	**0**	**1**	**0**	**1**
13	**1**	**1**	**0**	**0**
14	**1**	**1**	**0**	**1**

Saídas			
BR	BC	**AL**	Erro
0	0	**1**	0
0	0	**1**	0
0	1	**1**	0
0	0	**1**	0
0	1	**1**	0
0	0	**1**	0

As demais são falsas, ou seja, estão desligadas (0).

Mapas referente ao Alarme (Já agrupado)



Agora devemos extrair a expressão booleana correspondente a saída verdadeira (1) para o alarme (AL). Dois mapas foram montados para facilitar a ilustração, porém, deve ser montado um único mapa.

A expressão Booleana referente à saída *(AL) (primeiro MAPA)*

SNAR ⇒ Analisando as linhas onde o conjunto esta inserido, note que a entrada SNAR **não muda**, ou seja, na primeira linha é falso (0), na segunda linha é falsa (0). Neste caso, a saída somente será verdadeira (1) se SNAR ele for **invertido**.

SNBR ⇒ Ainda analisando as linhas onde o conjunto esta inserido, note que a entrada SNBR **muda**, ou seja, na primeira coluna é falsa (0) e na segunda é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

SNCR ⇒ Analisando as colunas onde o conjunto esta inserido, note que a entrada SC **não muda**, ou seja, na primeira coluna é falsa (0) e na segunda continua falsa (0). Neste caso a saída somente será verdadeira (1) para SNCR, se ele for **invertido**.

SC ⇒ Ainda analisando as colunas onde o conjunto esta inserido, note que a entrada SC **muda**, ou seja, na primeira coluna é falsa (0) e na segunda é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

A expressão Booleana referente à saída *(AL) (segundo MAPA)*

SNAR ⇒ Analisando as linhas onde o conjunto esta inserido, note que a entrada SNAR **muda**, ou seja, na primeira linha é falso (0), na segunda linha é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

SNBR ⇒ Ainda analisando as linhas onde o conjunto esta inserido, note que a entrada SNBR **não muda**, ou seja, na segunda linha é verdadeira (1) e na terceira é verdadeiro (1). Neste caso, a saída será verdadeira (1) sempre que SNBR for verdadeiro.

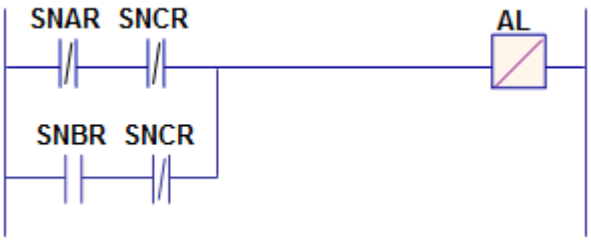
SNCR ⇒ Analisando as colunas onde o conjunto esta inserido, note que a entrada SC **não muda**, ou seja, na primeira coluna é falsa (0) e na segunda continua falsa (0). Neste caso, a saída somente será verdadeira (1) para SNCR, se ele for invertido.

SC ⇒ Ainda analisando as colunas onde o conjunto esta inserido, note que a entrada SC **muda**, ou seja, na primeira coluna é falsa (0) e na segunda é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

Cada conjunto representa uma função lógica "AND" e para juntar os dois conjuntos uma lógica "OR", assim, a saída AL só estará ligada (1) se SNAR e SNCR forem falsos (0) ou se SNBR for verdadeiro (1) e SNCR for falso (0) . Logo a expressão completa será:

AL = SNAR' . SNCR' + SNBR . SNCR'

Ladder correspondente ao alarme (AL)



Mapa indicador de erro (ERRO).

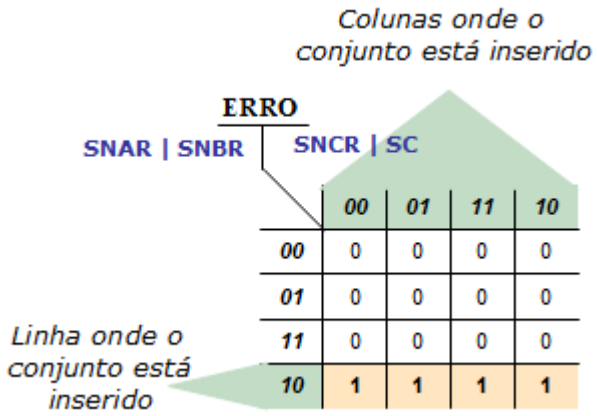
Na tabela verdade as condições em que o indicador de erro é verdadeira, ou seja, onde estará ligado, são:

	Entradas			
	SNAR	**SNBR**	**SNCR**	**SC**
08	**1**	**0**	**0**	**0**
09	**1**	**0**	**0**	**1**
10	**1**	**0**	**1**	**0**
11	**1**	**0**	**1**	**1**

Saídas			
BR	BC	AL	**Erro**
0	**0**	**0**	**1**
0	**0**	**0**	**1**
0	**0**	**0**	**1**
0	**0**	**0**	**1**

As demais são falsas, ou seja, estão desligadas (0).

Mapas referente ao ERRO (Já agrupado)



SNAR | SNBR \ SNCR | SC	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

A expressão Booleana referente à saída *(ERRO)*

SNAR ⇒ Como há somente uma linha onde o conjunto esta inserido a entrada SNAR **não muda**, ou seja, é verdadeiro (1). Neste caso, a saída será verdadeira (1) sempre que SNAR for verdadeiro.

SNBR ⇒ Como também há somente uma linha onde o conjunto esta inserido a entrada SNBR **não muda**, ou seja, é falso (0). Neste caso, a saída somente será verdadeira (1) para SNCR, se ele for invertido.

SNCR ⇒ Analisando as colunas onde o conjunto esta inserido, note que a entrada SNCR **muda**, ou seja, na primeira coluna é falsa (0), na segunda continua falsa (0) porém na terceira é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

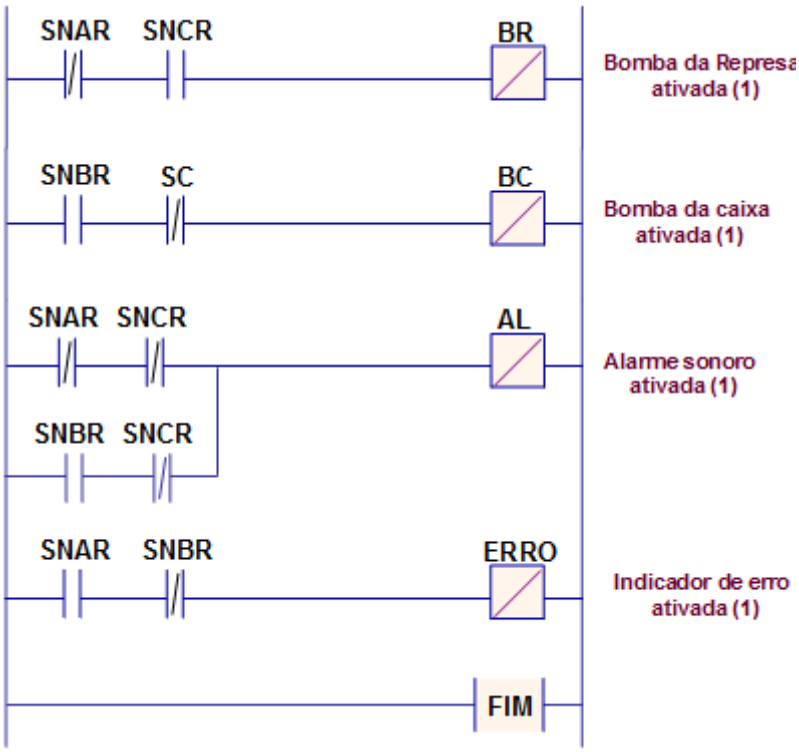
SC ⇒ Analisando as colunas onde o conjunto esta inserido, note que a entrada SC **muda**, ou seja, na primeira coluna é falsa (0), na segunda é verdadeiro (1). Neste caso, esta entrada deve ser ignorada.

Cada conjunto representa uma função lógica “AND”, assim, a saída ERRO só estará ligada (1) se SNAR for verdadeiro e SNCR invertido. Logo a expressão é:

ERRO = SNAR .SNBR’

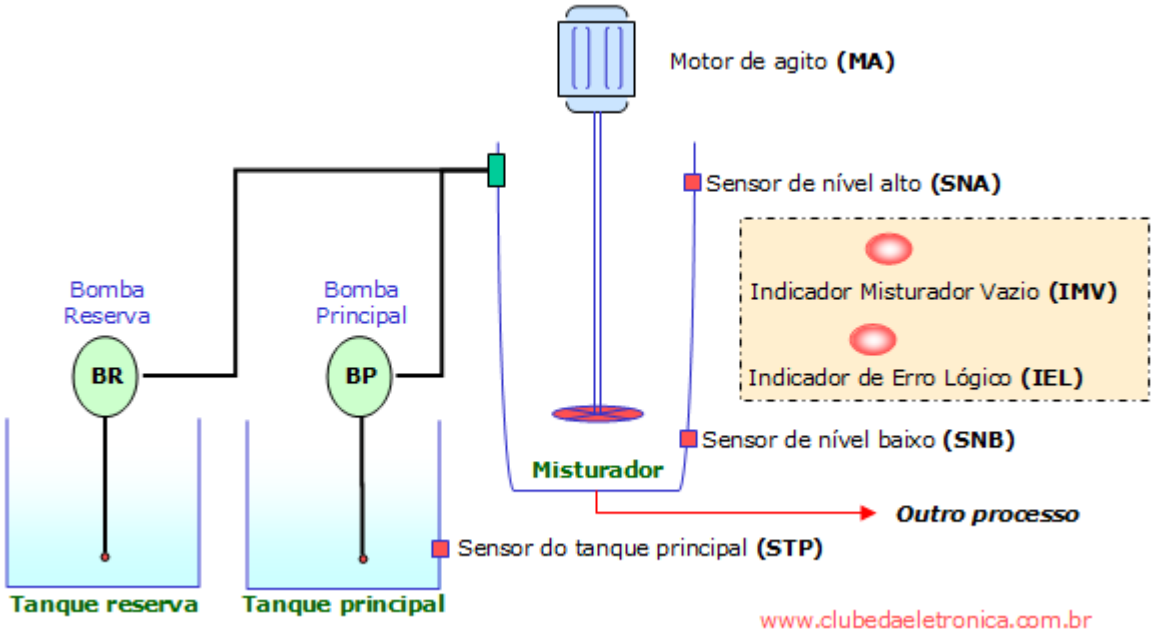
Ladder correspondente ao indicador de erro (ERRO)

Ladder completo:



Os mapas não são recomendados em sistemas com mais de 4 entradas, se for necessário as máquinas de estados são mais adequadas.

Praticando...
Dado o sistema abaixo, elabore um sistema lógico, em ladder, que atenda todos os critérios requeridos.



Critérios:

- O misturador deverá estar sempre cheio, ou seja, **SNA=1**. Se **SNA=0**, a bomba principal **(BP)** deverá ser acionada (1), mas somente se houver produto no tanque principal **(STP=1)**, se não houver **(STP=0)** a bomba reserva **(BR)** deve ser acionada (1).
- Sempre que o sensor de nível baixo **(SNB)** estiver em 0, um indicador **(IMV)** deverá avisar ao operador que o misturador está vazio.
- Se ocorrer um erro lógico (situação impossível), todas as saídas deverão ser desligadas e o operador deverá ser avisado através do indicador de erro lógico **IEL**.
- Se o misturador estiver cheio, o motor de agito **(MA)** deve ser acionado.

Etapas à seguir:

1- Definir as entradas e saídas

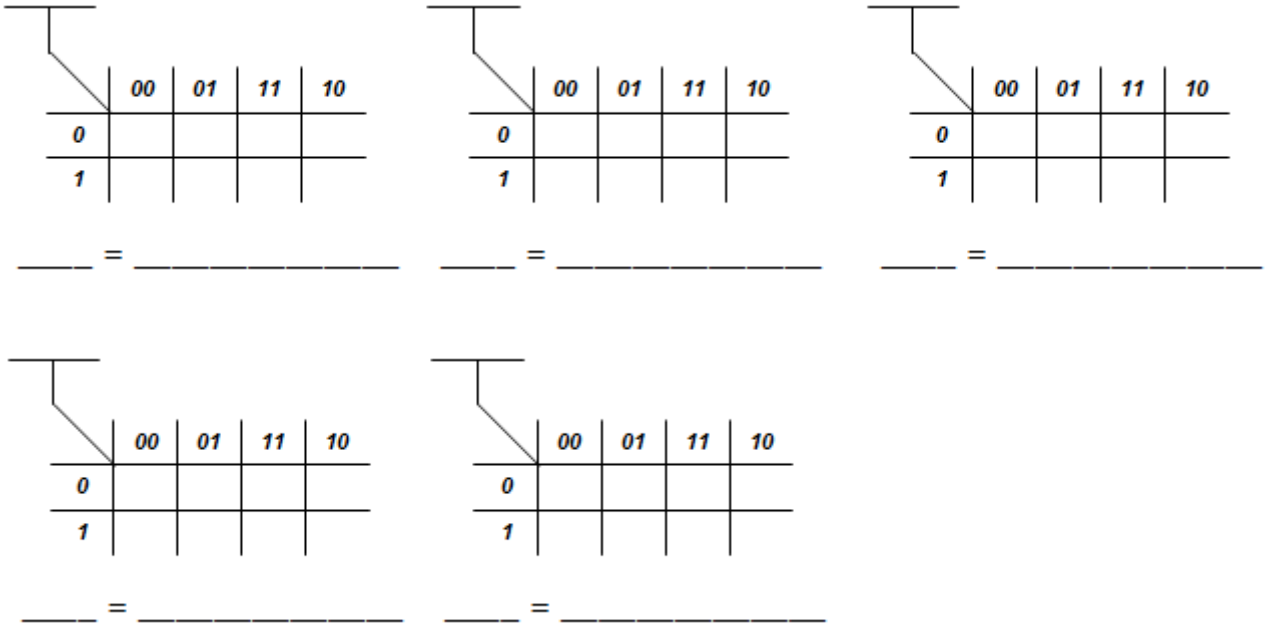
Entradas		Saídas	

Ilustração:
2- Elaborar tabela verdade

Entradas		

saídas				

3- Elaborar os mapas e extrair as expressões lógicas



4- Elaborar o ladder e testar

2 - Sistema de votação

Deseja-se implementar um sistema lógico simplificado para um sistema de votação de uma empresa, que tem sua diretoria constituída pelos seguintes elementos: **D**iretor, **V**ice-diretor, **S**ecretário e **T**esoureiro.

Uma vez por mês esta diretoria se reúne para discutir sobre os mais diversos assuntos, sendo que as propostas são ou não **A**ceitas. Devido o número de elementos da diretoria ser par, o sistema adotado é o seguinte:

- Maioria → A proposta é aceita
- Minoria → A proposta é rejeitada
- Empate → Vence o voto dado pelo diretor

Etapas à seguir:

a. Complete a tabela verdade de maneira que atenda as exigências.
b. Complete o mapa para simplificação.
c. Extraia e expressão lógica.
d. Elabole e teste o programa em ladder.

"Olhe à frente, para que a vista preceda os passos."

Salomão, rei de Israel.

www.clubedaeletronica.com.br

Referências bibliográficas:

- Circuitos digitais, Autor: Antonio Carlos de Oliveira Lourenço, Ed. Érica.
- http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/
- http://www.cpdee.ufmg.br/~carmela/NORMA%20IEC%201131.doc
- http://www.software.rockwell.com/corporate/reference/Iec1131/
- http://www.plcopen.org/
- http://www.lme.usp.br/~fonseca/psi2562%20aula%206%20IHM.pdf
- http://www.teses.usp.br/teses/disponiveis/18/18133/tde-11072002-085859/
- http://www.redenet.edu.br/publicacoes/arquivos/20080108_144615_INDU-058.pdf
- http://www.corradi.junior.nom.br/modCLP.pdf
- http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/

Parte 06 - Técnicas de programação (máquina de estados)

MÁQUINA DE ESTADOS FINITO

Pode-se definir máquina de estado como sendo um modelo de comportamento de um determinado processo, em nosso caso industrial. Uma máquina de estado é composta por estados, transições e saídas.

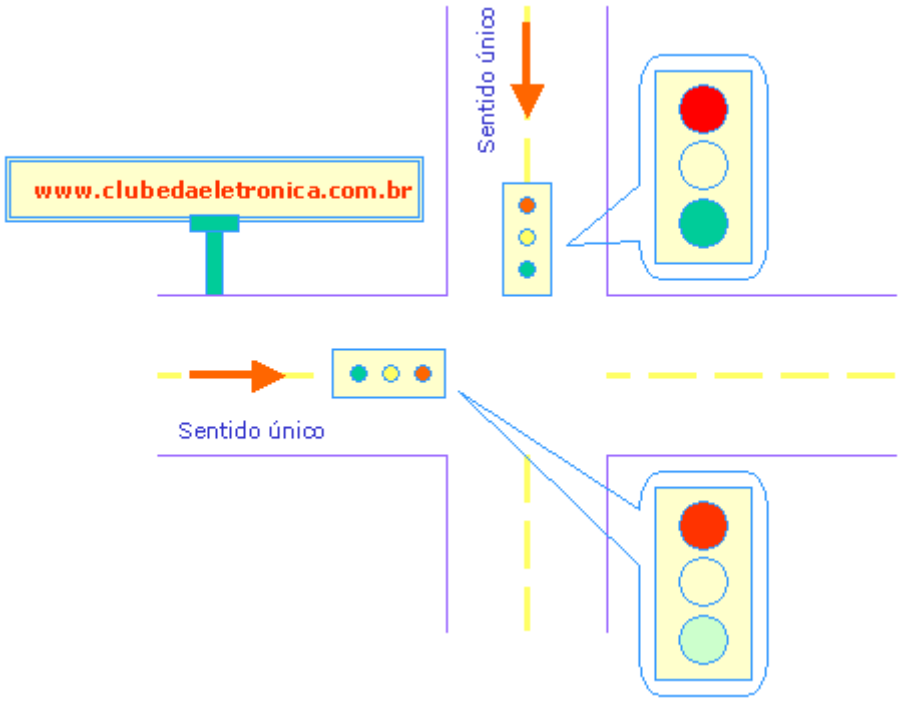
- **Estado** → comporta-se como uma memória, ou seja, armazena todas as informações sobre as saídas em um determinado momento.
- **Transição** → é a condição para que ocorra a mudança de um estado para outro.
- **Saída** → descreve a atividade que deve ser realizada num determinado estado.

A máquina de estado é representada por um diagrama bastante simplificado, conhecido como diagrama de transição de estado, que tem como objetivo facilitar o entendimento de qualquer pessoa interessada no processo.

O objetivo deste material é demonstrar com exemplos como a máquina de estado reproduz fielmente todas as etapas idealizadas pelo projetista.

Sistema seqüencial simples – controle de tráfego (Resolvido)

Para que um programa faça o que você quer, você deve dizer o que ele deve fazer, sem omitir qualquer passo. Para explicar a máquina de estado primeiramente faremos um sistema lógico simples e conhecido por todos, trata-se de um controle de trafego. A figura abaixo ilustra a idéia proposta.



Descrição de funcionamento

O semáforo terá início quando um botão (BL) tipo “push button” for acionado, dar-se-à então início ao ciclo. O tempo (T) ficará a critério do operador e o sistema poderá ser desligado em qualquer momento através de um botão (BD), também push button.

Definição de entradas e saídas (I/O)

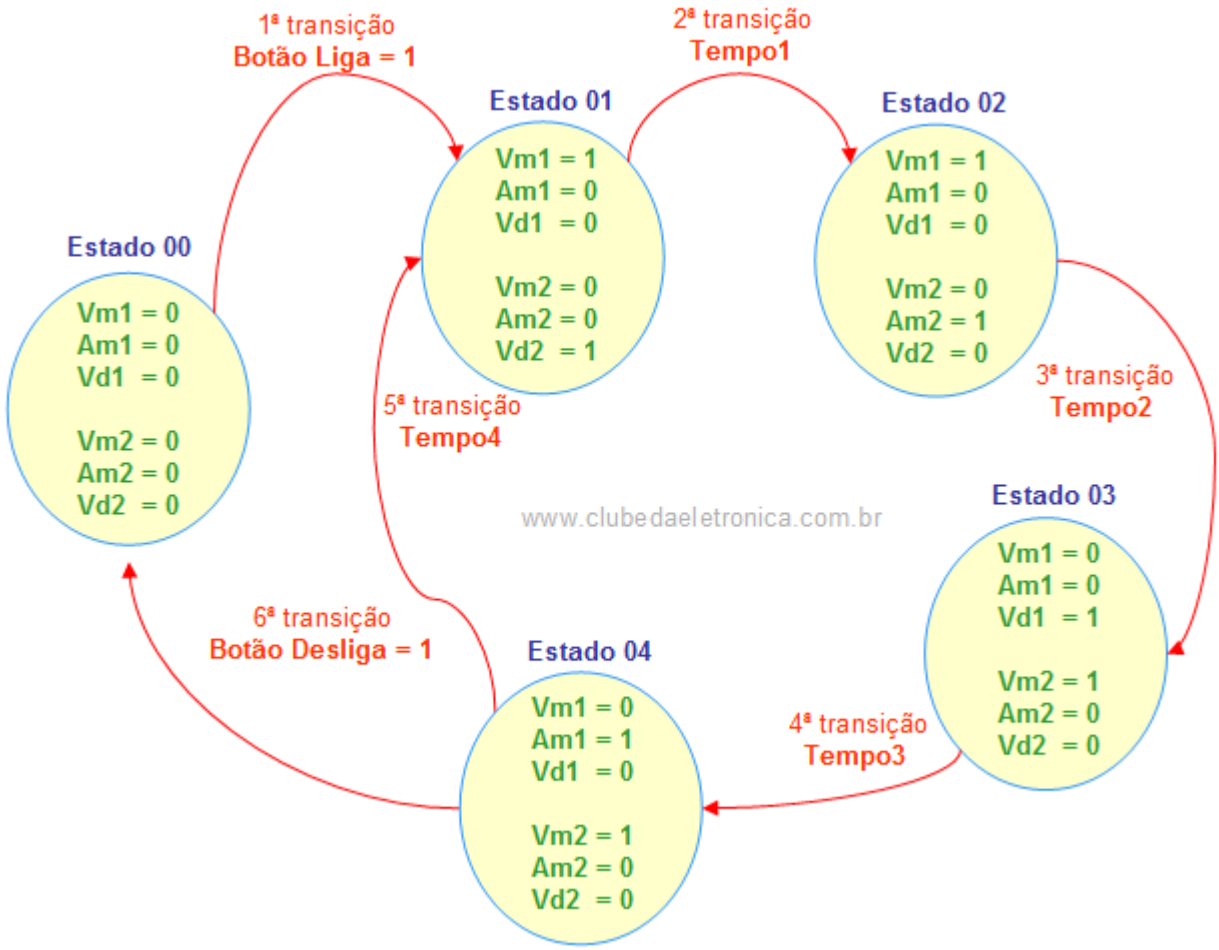
Entradas e saídas (E/S) ou input/output (I/O) são termos comuns na automação, isso porque, em todo sistema há necessidade da inserção de informações (I), que devem se processadas através de uma unidade central de processamento (CPU) e enviadas às saídas (O), para que uma determinada ação, previamente programada, seja tomada.

Lista de entradas e saídas (I/O) para o semáforo

Entradas	Saídas Semáforo 1	Saídas Semáforo 2
Botão liga = I0	Vermelho 1 = O0	Vermelho 2 = O3
Botão desliga = I1	Amarelo 1 = O1	Amarelo 2 = O4
	Verde 1 = O2	Verde 2 = O5

Elaboração da máquina de estado

O objetivo da máquina de estados é modelar através do diagrama de estado o comportamento do projeto, no caso, o controle de trafego. Sua representação é feita com detalhes a fim de facilitar o entendimento. Abaixo, sua representação já aplicada ao semáforo.



Explicação detalhada

Estado 00 ⇒ Representa o estado inicial da máquina, ou seja, todas as saídas estão desligadas.

1ª Transição ⇒ Para que o sistema mude de um estado para outro, é necessário que haja a incersão de informações, em nosso caso, estado 01 só será ligado se BL for acionado.

Estado 01 ⇒ Representa o estado após o recebimento da informação, neste caso, as lâmpadas vermelho 01 e Verde 02 estarão em nível alto, ou seja, acesas enquanto que as outras estarão apagadas.

2ª Transição ⇒ O estado 01 permanecerá até que termine o tempo previamente estabelecido pelo projetista, neste caso, o tempo representa uma transição e por conseqüência uma entrada.

Estado 02 ⇒ Ao terminar o tempo estabelecido na 2ª transição, haverá mudança do estado 01 para o estado 02 e assim, suas saídas serão acionadas, neste caso, as lâmpadas vermelho 01 e amarelo 02 estarão em nível alto, ou seja, acesas enquanto que as outras estarão apagadas.

3ª Transição ⇒ Mais um tempo é necessário para que ocorra a mudança de estado, ou seja, só mudará do estado 02 para o estado 03 no final do se o podendo , O estado 01 permanecerá até que termine o tempo estabelecido no programa.

Estado 03 ⇒ Ao final da 2º transição, o estado 03 é acionado, e com ele as lâmpadas verde 01 e vermelho 02 estarão em nível alto, ou seja, acesas enquanto que as outras estarão apagadas.

4ª Transição ⇒ Mais um tempo é necessário para que ocorra a mudança de estado, ou seja, só mudará do estado 03 para o estado 04 no final do tempo estabelecido no programa.

Estado 04 ⇒ Ao final da 3º transição, o estado 04 é acionado, e com ele as lâmpadas amarelo 01 e vermelho 02 estarão em nível alto, ou seja, acesas enquanto que as outras estarão apagadas.

5ª Transição ⇒ Um último tempo é necessário para que o processo reinicie, ou seja, volta ao estado 01 e o ciclo recomeça.

6ª Transição ⇒ A 6ª transição tem função de parar o sistema. Note que todos os estados serão desligados simultaneamente.

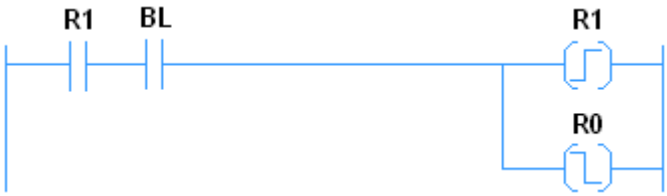
O Programa ladder

O programa ladder deve ser fiel a máquina de estados e se esta última for perfeita o programa também será.

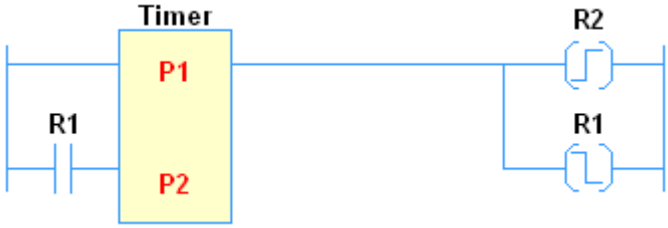
Detalhes da lógica Ladder

Lógica 01 ⇒ Cada estado de seu sistema deve ser associado a um contato auxiliar **(R)**. Como na máquina há quatros estados, temos assim, quatro contatos auxiliares, ligando o estado inicial (tudo desligado).

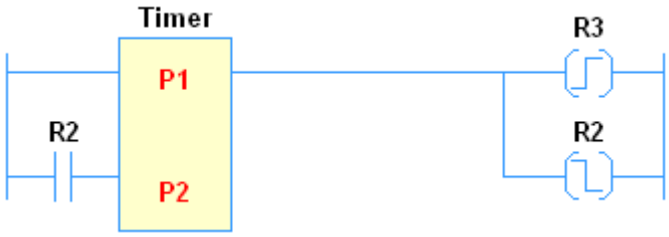
Lógica 02 ⇒ Se o estado inicial **(R0)** estiver acionado e se o Botão (push button) liga **(BL)** for pressionado é setado o estado **(R1)** e resetado o estado **(R0)**.



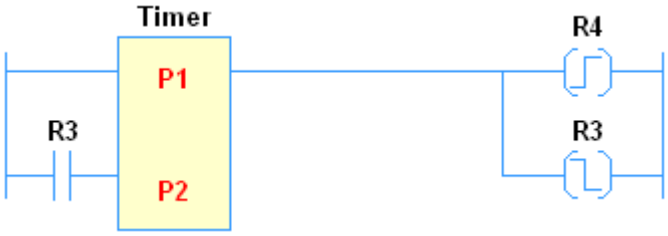
Lógica 03 ⇒ Se o estado **(R1)** estiver setado dar-se-á inicio a contagem de tempo **(T1)** e ao final o estado **(R2)** será setado e **(R1)** ressetado.



Lógica 04⇒ Se o estado **(R2)** estiver setado dar-se-á inicio a contagem de tempo **(T2)** e ao final o estado **(R3)** será setado e **(R2)** ressetado.

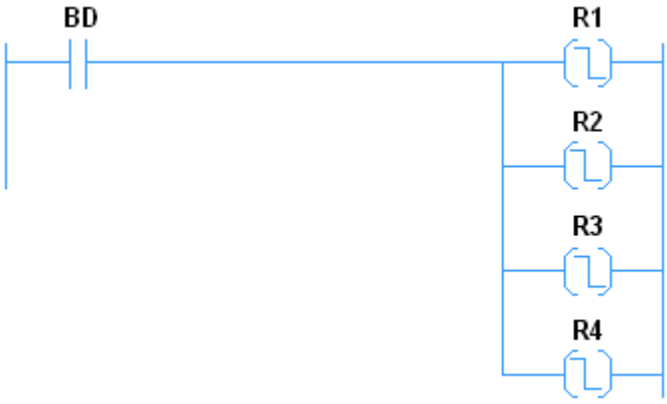


Lógica 05 ⇒ Se o estado **(R3)** estiver setado dar-se-á inicio a contagem de tempo **(T3)** e ao final o estado **(R4)** será setado e **(R3)** ressetado.

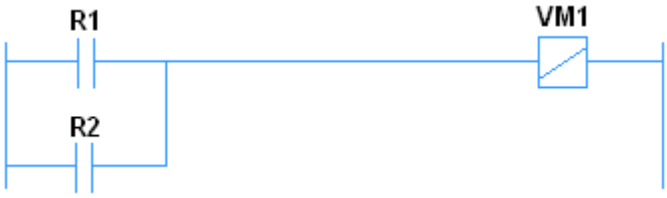


Lógica 06 ⇒ Se o estado **(R4)** estiver setado dar-se-á inicio a contagem de tempo **(T4)** e ao final o estado **(R1)** será setado e **(R4)** ressetado. O ciclo recomeça.

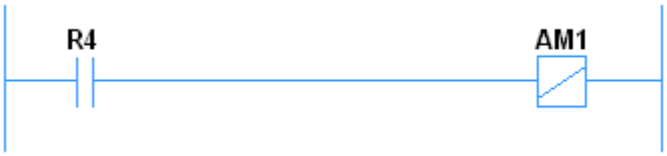
Lógica 07 ⇒ Se o botão (push button) for pressionado todos os estados **(R1)**, **(R2)**, **(R3)** e **(R4)** serão ressetados.



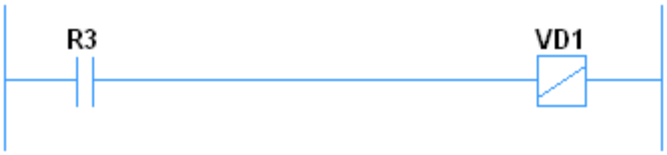
Lógica 08 ⇒ Acionando a saída vermelha do semáforo 01 **(VM1)**. Observando a máquina de estado note que a saída vermelho 01 está em nível lógico alto nos estados **(R1)** e **(R2)**. Assim, os estados **(R1)** ou **(R2)** devem acionar a saída **(VM1)**.



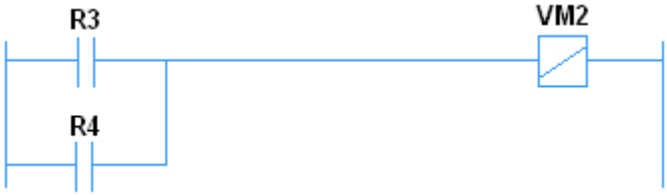
Lógica 09 ⇒ Acionando a saída amarela do semáforo 01 **(AM1)**. Observando a máquina de estado note que a saída amarela 01 está em nível lógico alto somente no estado **(R4)**. Assim, o estado R4 deve acionar a saída **(AM1)**.



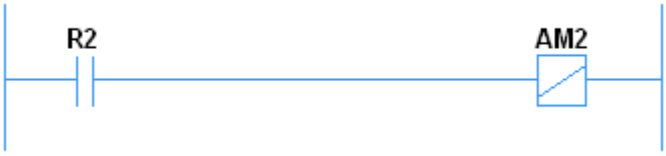
Lógica 10 ⇒ Acionando a saída verde do semáforo 01 **(VD1)**. Observando a máquina de estado note que a saída verde 01 está em nível lógico alto somente no estado **(R3)**. Assim, o estado R3 deve acionar a saída **(VD1)**.



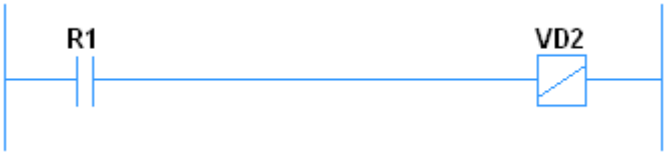
Lógica 11 ⇒ Acionando a saída vermelha do semáforo 02 **(VM2)**. Observando a máquina de estado note que a saída vermelha 02 está em nível lógico alto nos estados **(R3)** e **(R4)**. Assim, o estado os estados **(R3)** ou **(R4)** devem acionar a saída **(VM2)**.



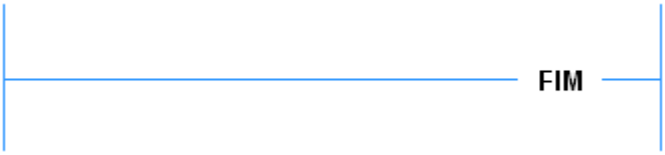
Lógica 12 ⇒ Acionando a saída amarela 02 do semáforo 02 **(AM2)**. Observando a máquina de estado note que a saída amarela 02 está em nível lógico alto somente no estado **(R2)**. Assim, o estado o estado **(R2)** deve acionar a saída **(AM2)**.



Lógica 13 ⇒ Acionando a saída verde 02 do semáforo 02 **(VD2)**. Observando a máquina de estado note que a saída amarela 02 está em nível lógico alto somente no estado **(R1)**. Assim, o estado o estado **(R1)** deve acionar a saída **(VD2)**.

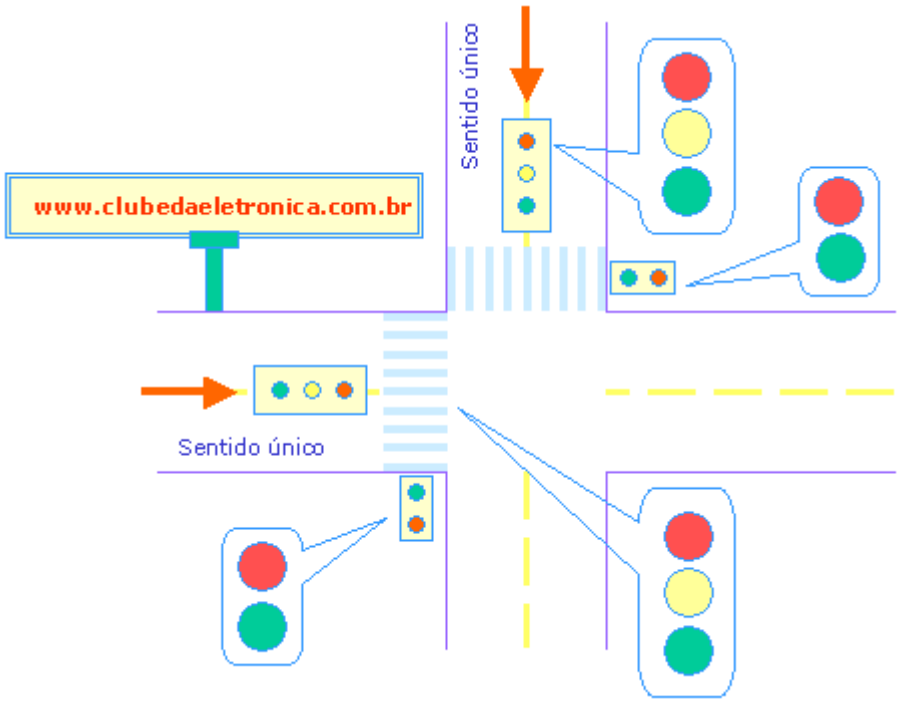


Lógica 14 ⇒ Fim de programa.



Praticando...

Deseja-se implementar um sistema de controle para semáforos para veículos e pedestres, como mostrado no esquema:



Descrição de funcionamento:

O circuito é iniciado quando o **botão de Liga (BL)** tipo “push button” é acionado, dar-se-á então, o início ao ciclo. O tempo fica a critério do projetista tomando cuidado para que não ocorram choques ou atropelamentos. Um **botão de desliga (BD)** tipo “push button” também deve ser implementado.

Etapas para relatório do projeto

- Descrição funcional (apontar possíveis falhas no processo e soluciona-las);
- Identificar as variáveis e criar a lista de entradas e saídas;
- Fazer o diagrama de estado da solução proposta;
- Implementar no Kit do Zap500 (não esquecer dos comentários); e

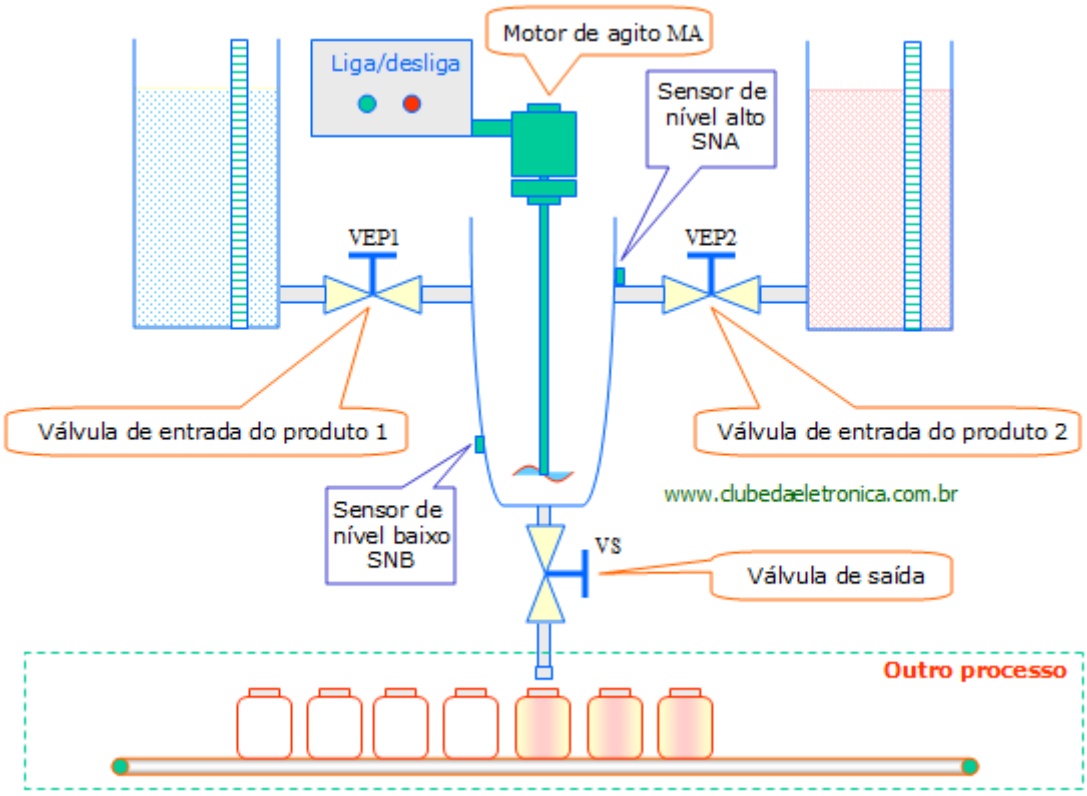
Mensagens na IHM do CLP

Hoje, praticamente todos os fabricantes de CLP (Controladores lógicos programáveis) disponibilizam a IHM (Interface Homem Máquina) como acessório opcional que, diga-se de passagem, agrega bastante valor ao CLP.

Normalmente a IHM consiste de um teclado para entrada de dados e uma tela (display) para visualização das informações pertinentes ao sistema. Abaixo, um exemplo resolvido ilustra o uso das mensagens na IHM.

Agitador de produtos (Resolvido)

Deseja-se implementar o sistema de controle do esquema abaixo:



Descrição de funcionamento

Ao pressionar o botão liga (**BL**), dar-se-á início ao processo abrindo simultaneamente as válvulas de entrada **VEP1** e **VEP2**; quando o sensor de nível alto **(SNA)** for atingido, automaticamente as válvulas de entrada se fecham e o motor agitador funciona por 10 segundos; A válvula de saída **(VS)** será aberta e o

líquido escoará até que o sensor de nível baixo **(SNB)** seja atingido, então o ciclo recomeça. Pressionando o botão desliga (**BD**) o processo será interrompido.

Etapas para elaboração do projeto

- Definir o problema corrigindo possíveis falhas no processo.
- Identificar as variáveis e criar a lista de entradas e saídas.
- Montar o diagrama de estado da solução proposta.
- Implementar no Kit do Zap500 (não esquecer dos comentários e identificação das variáveis)

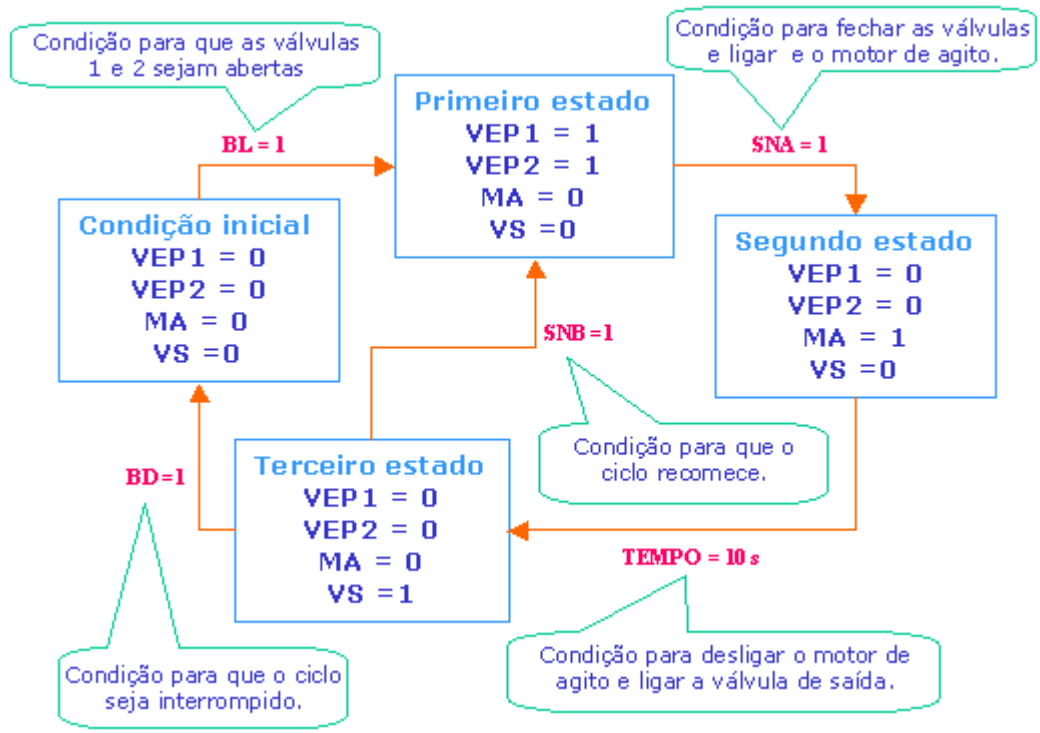
Solução:

Definição de entradas e saídas (I/O)

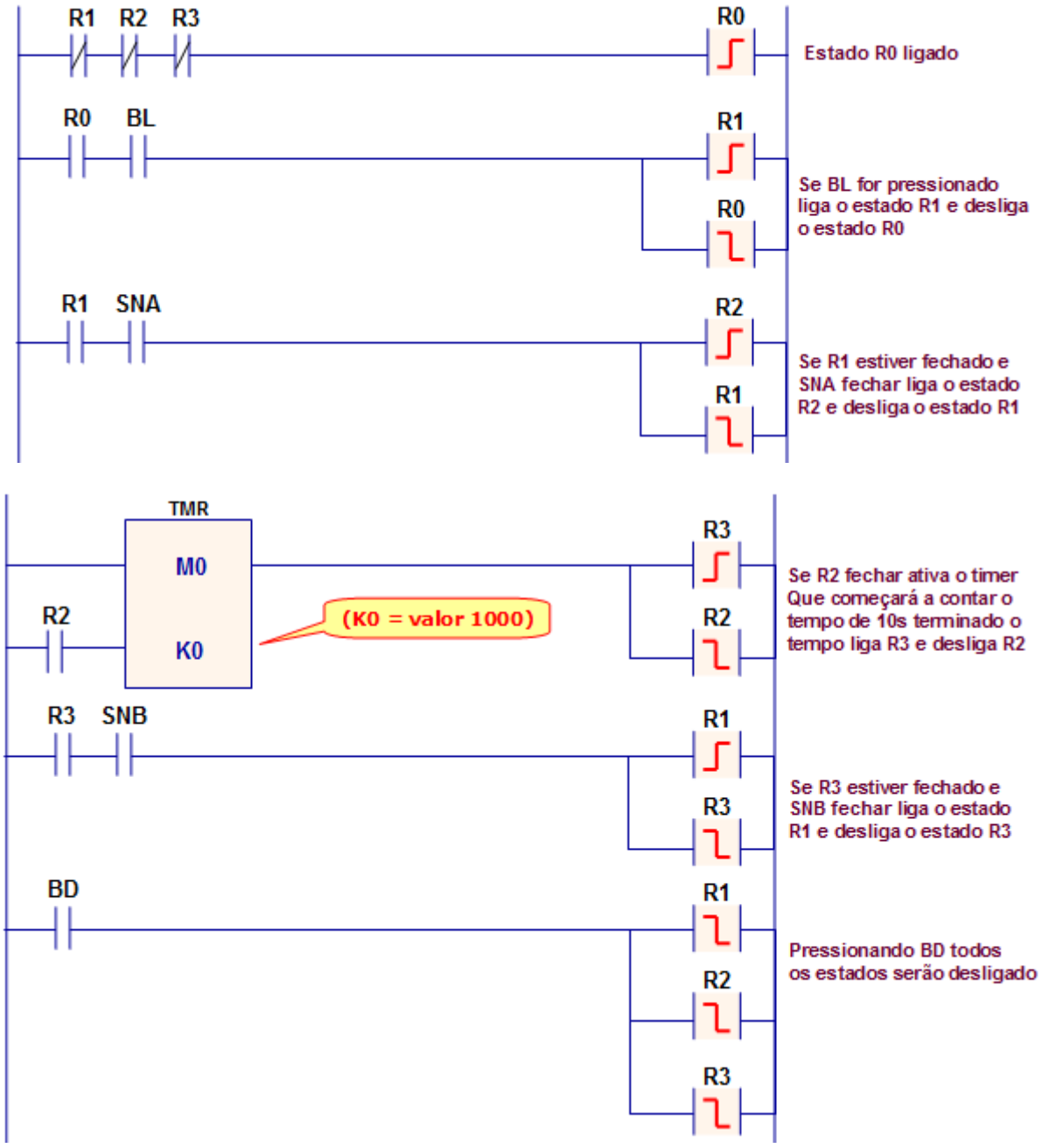
As entradas são as condições (transições) impostas pelo projetista para ocorra uma mudança de estado, e as saídas são os atuadores, ou seja, os que executarão algo se uma determinada condição for atingida. No projeto a ser implementado as entradas e as saídas, são:

Entradas	Saídas
✓ **BL** = Botão de liga	✓ **VEP1** = Válvula de entrada do produto 1
✓ **BD** = Botão de desliga	✓ **VEP2** = Válvula de entrada do produto 2
✓ **SNA** = Sensor de nível alto	✓ **MA** = Motor agitador
✓ **SNB** = Sensor de nível baixo	✓ **VS** = Válvula de saída.

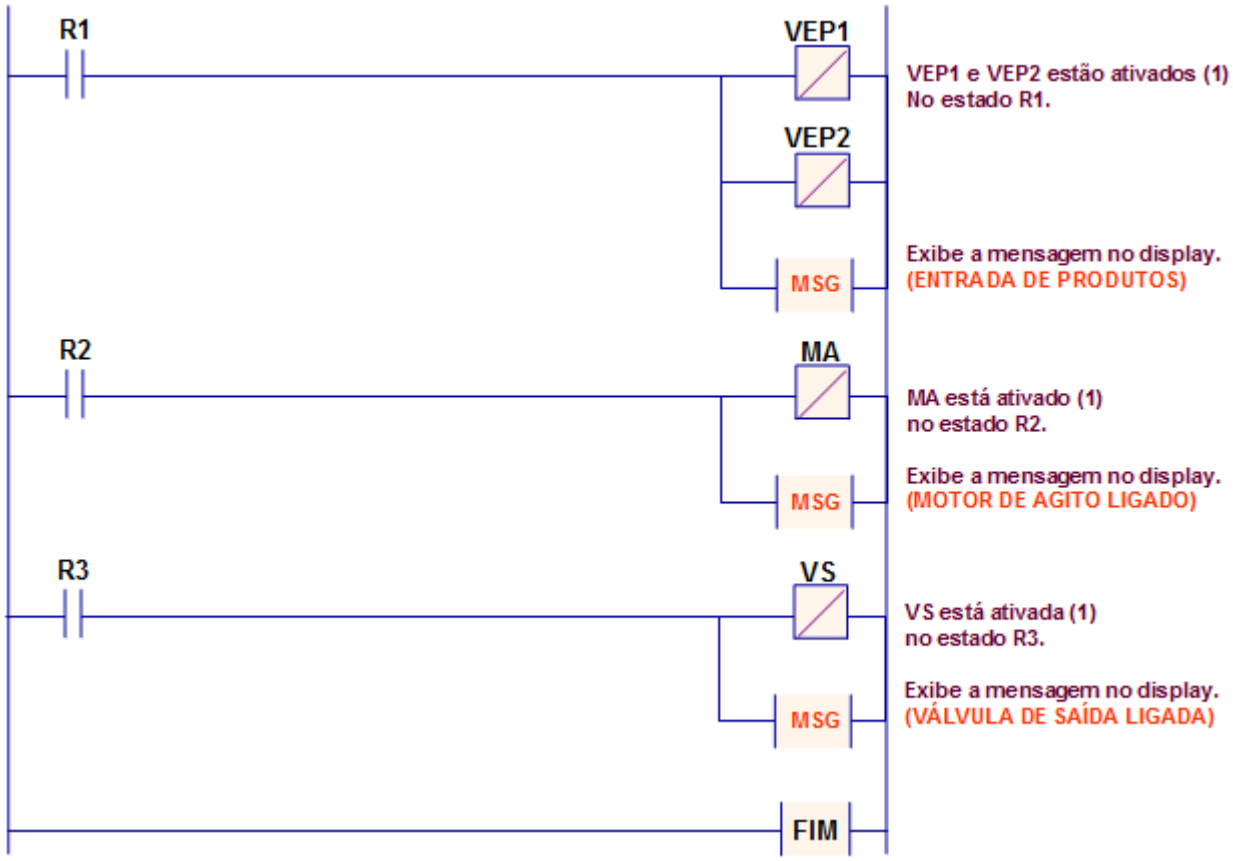
Máquina de estados



O programa ladder com comentários

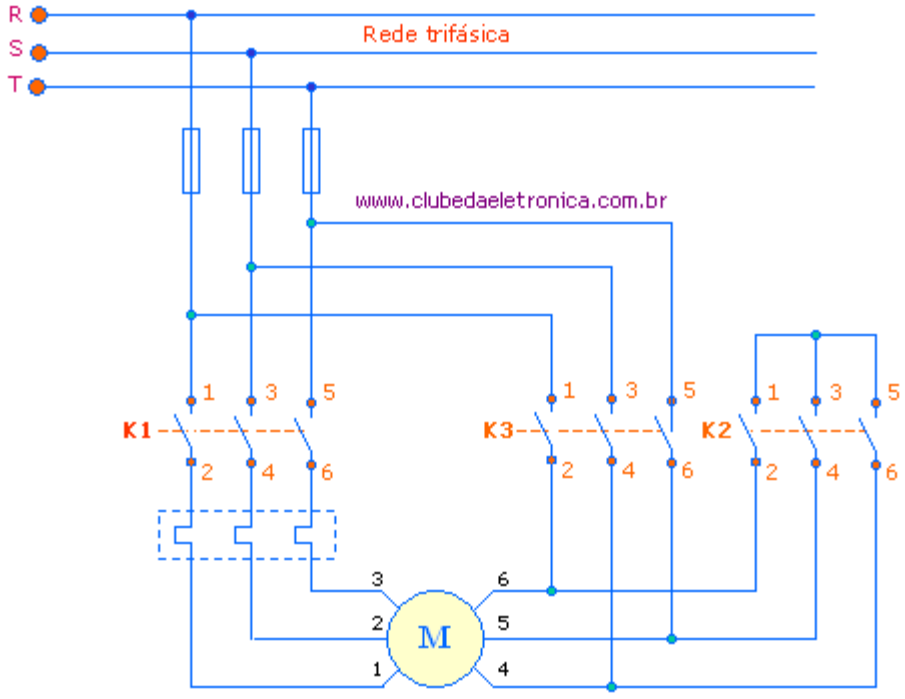


Energizando as saídas.

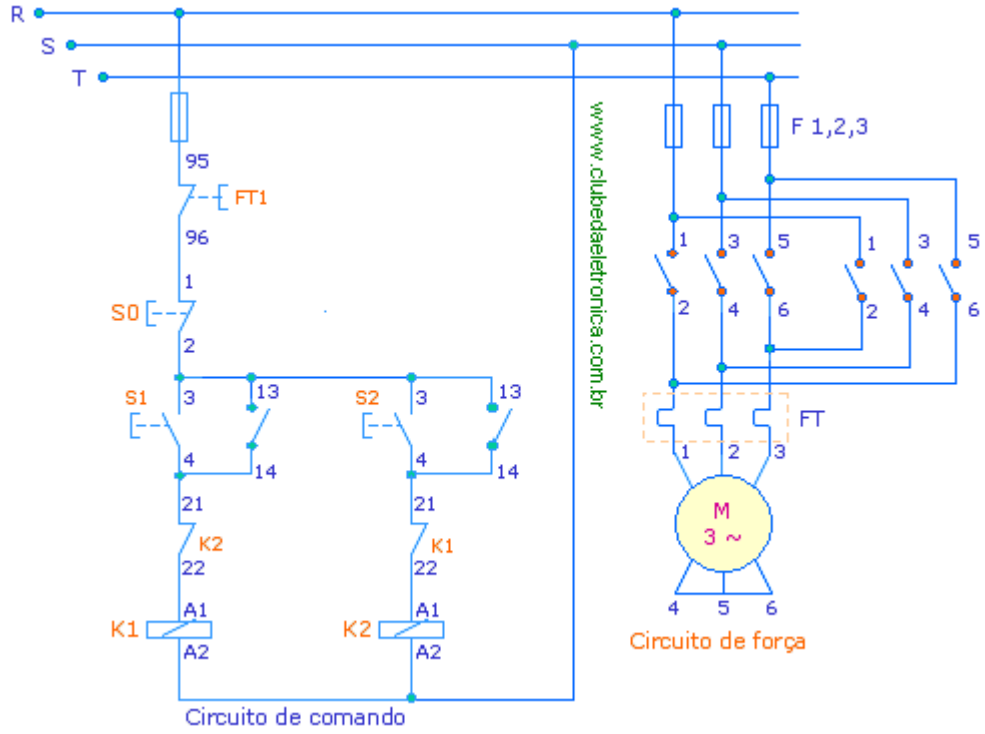


Praticando...

Elabore em ladder um programa capaz de partir o motor em Y (exibir mensagem TENSÃO 127V) e após 3 segundos comutá-lo automaticamente para Δ (exibir mensagem TENSÃO 220V). O diagrama de força esta representado abaixo.



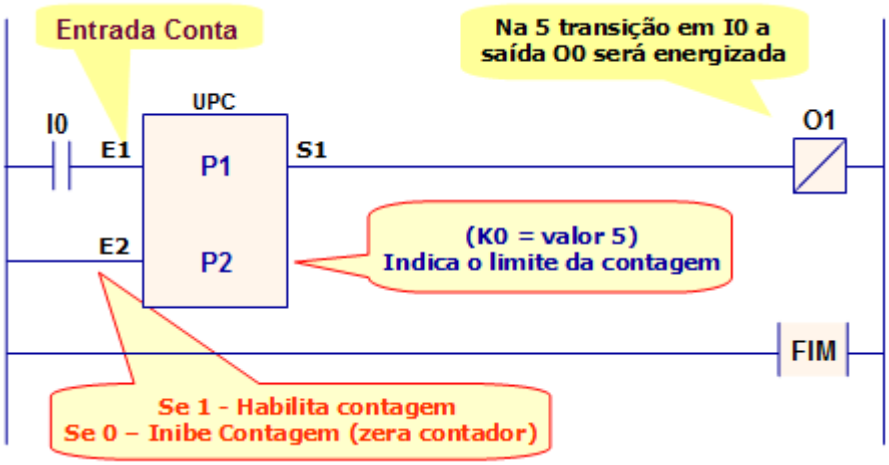
Elabore em ladder uma partida direta reversa, exibindo as seguintes mensagens (SENTIDO HORÁRIO e SENTIDO ANTI-HORÁRIO). O diagrama de comando e de força está representado abaixo.



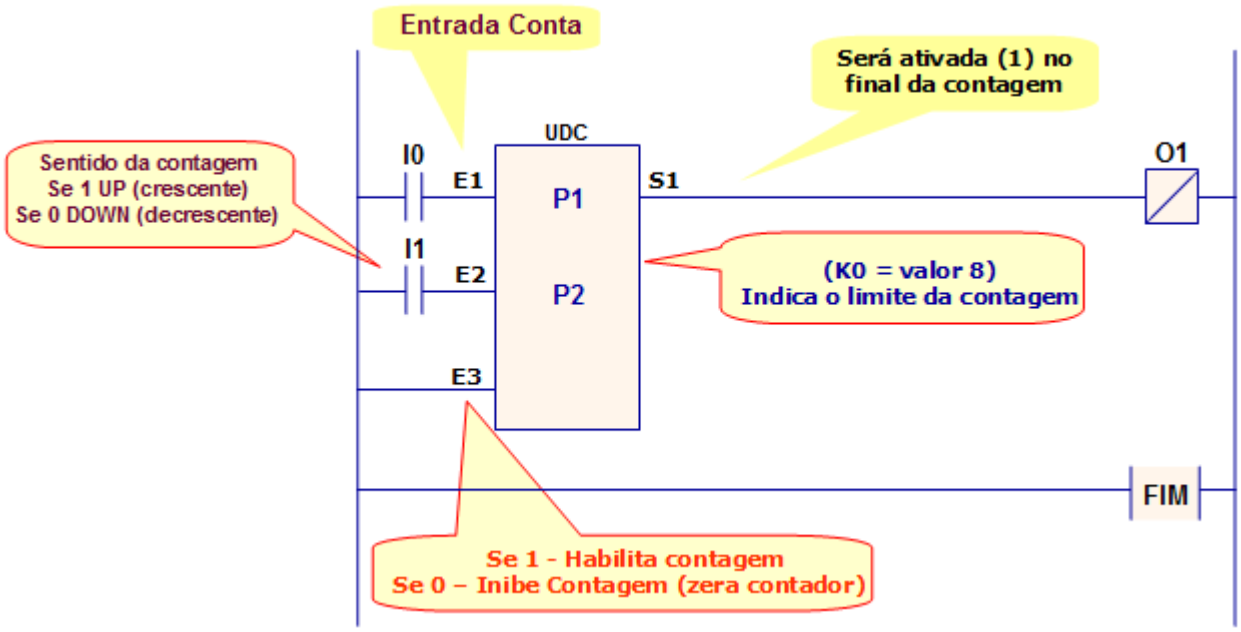
Blocos contadores UP (crescente) e UP/DOWN (crescente e decrescente)

São blocos destinados à contar um determinado número de transições ocorridas na entrada “conta”. Ele conta o número de transições até um valor fornecido pelo usuário como parâmetro. A saída indica o fim da contagem. Os contadores podem ser UP e UP/DOWN, vejamos:

Contador UP (crescente)



Contador UP (crescente)/DOWN(decrescente)



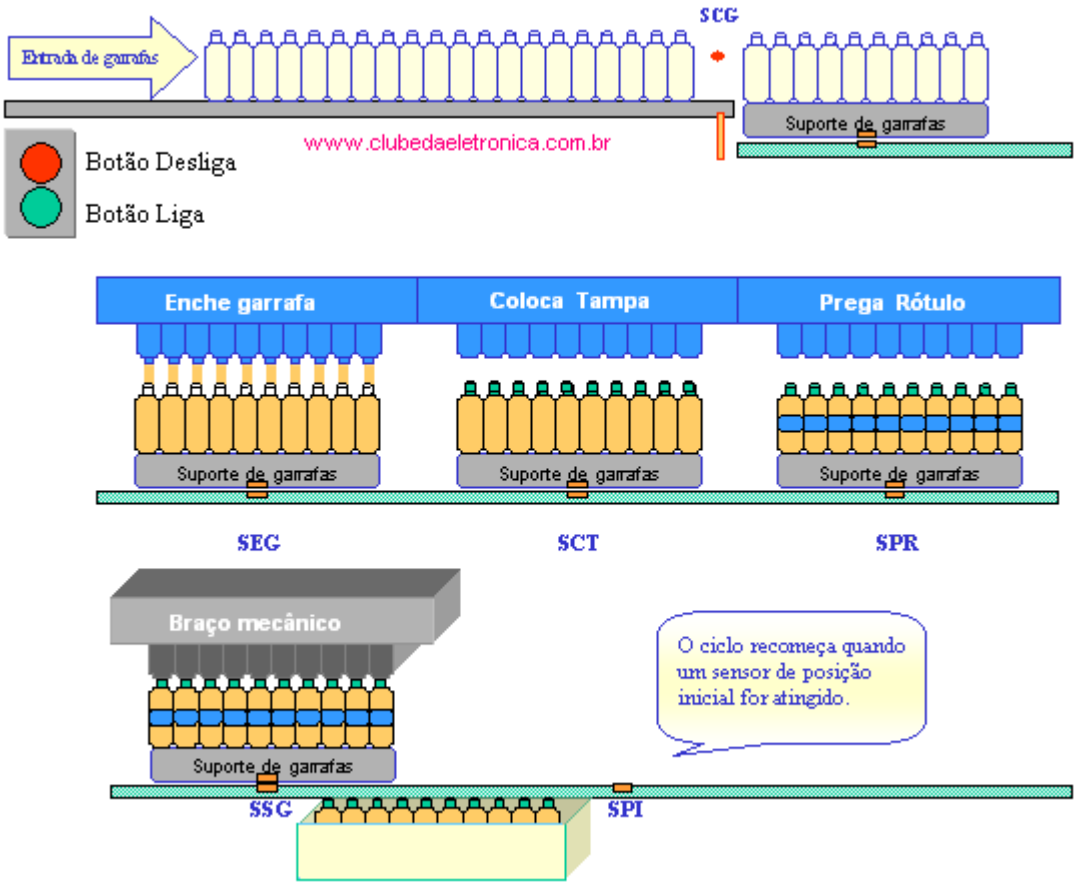
Os parâmetros P1 e P2 são iguais para ambos os contadores sendo:

P1 – representa o valor corrente da contagem e deve ser obrigatoriamente uma memória inteira (M).

P2 – representa o valor limite da contagem e pode ser memória inteira (M) ou constante inteira (K).

Praticando...

O projeto abaixo tem o objetivo didático, onde o aluno deverá entender o sistema descrito e automatizá-lo.



Descrição de funcionamento

- Pressionando o botão liga **(BL)** as garrafas começarão a entrar (processo não descrito) por uma esteira secundária **(EG) .**
- Haverá um controle de garrafas que será colocada sobre o suporte, em cada suporte haverá 10 garrafas, que serão contadas por intermédio de um sensor **(SCG)**.
- Assim que as dez garrafas estiverem no suporte, a esteira se movimentará através de um motor **(ME)**;
- O motor da esteira irá parar assim que o suporte atingir o sensor de enchimento **(SEG)**;
- Atingido o **SEG**, deverá acionar o motor de controle de descida da máquina que encherá as garrafas **(MEG)** que deverá ficar baixo por 5 segundos (tempo para enchimento da garrafa) e retornará a sua posição de origem aguardando próximo suporte.
- Assim o motor da esteira ligará novamente até chegar no sensor de tampa **(SCT)** que deverá acionar o motor de controle de descida da segunda máquina que será responsável por colocar a tampa esse processo levará 2 segundos e retornará a sua posição de origem aguardando próximo suporte.
- Novamente a esteira será ligada e chegará o sensor de prega de rótulo **(SPR)** que deverá acionar o motor de controle de descida da segunda máquina que será responsável pela colocação do rotulo esse processo levará 3 segundos e retornará a sua posição de origem aguardando próximo suporte.
- Assim que o rótulo for colocado a esteira se movimentará novamente até encontrar um sensor de saída de garrafas **(SSG)**, onde um braço mecânico irá retirar as garrafas do suporte, para coloca-las em uma caixa. O tempo de retirada será de aproximadamente 3s.
- A esteira liga novamente levando o suporte para a origem **(SPI)** que aguardará outras garrafas.

Importante:

- O sistema deverá ser ligado por intermédio de um botão de liga **(BL)** e desligado através de um botão desliga **(BD)**. **(Usar IHM)**
- Assim que o sistema for ligado as garrafas serão inseridas em um suporte, dando inicio a contagem.
- O processo acima é passivo pode ser melhorado, aceitando sugestões do projetista.
- Verifique se há erros no processo e aponte soluções.

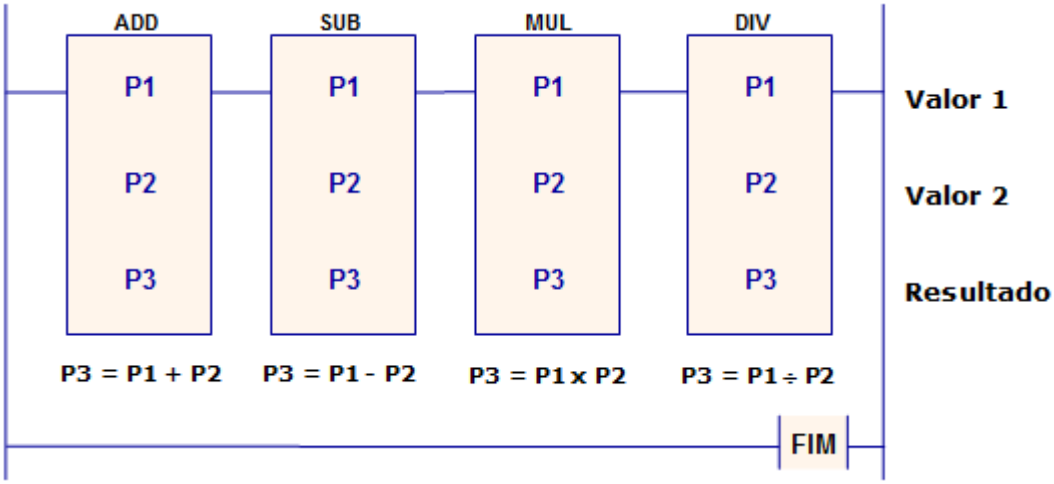
Etapas à seguir:

a) Identificar as variáveis e criar a lista de E/S.
b) Fazer os diagramas de estado, fluxograma, etc. da solução proposta.
c) Implementar no Kit do Zap500 (utilizar IHM para monitorar saídas)

Blocos matemáticos

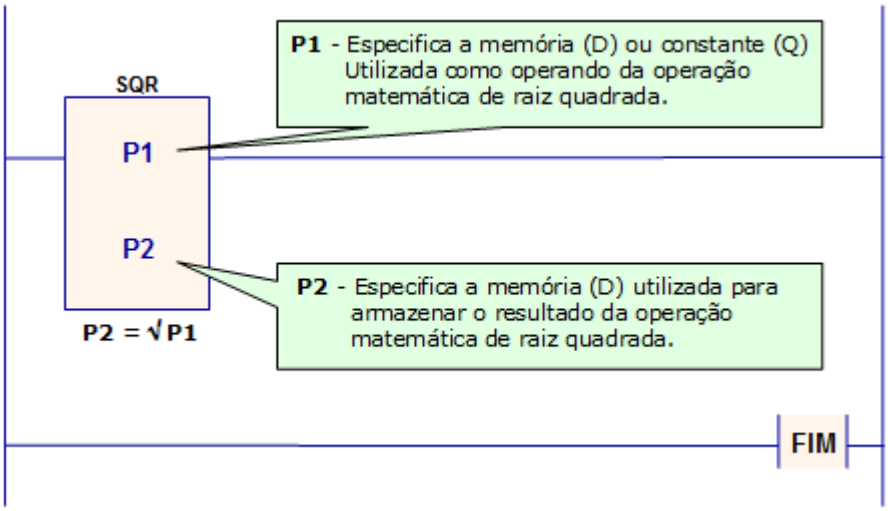
Como o próprio nome diz estes blocos realizam operações matemáticas como Adição (**ADD**), Subtração **(SUB)**, Multiplicação **(MUL)** e Divisão **(DIV)**.

As operações são realizadas entre os operandos P1 e P2 que podem ser Memória inteira **(M)**, Memória real **(D)**, Constante inteira **(K)**, ou constante real **(Q)** armazenando o resultado em P3 que pode ser uma memória Inteira **(M)** ou real **(D)**.

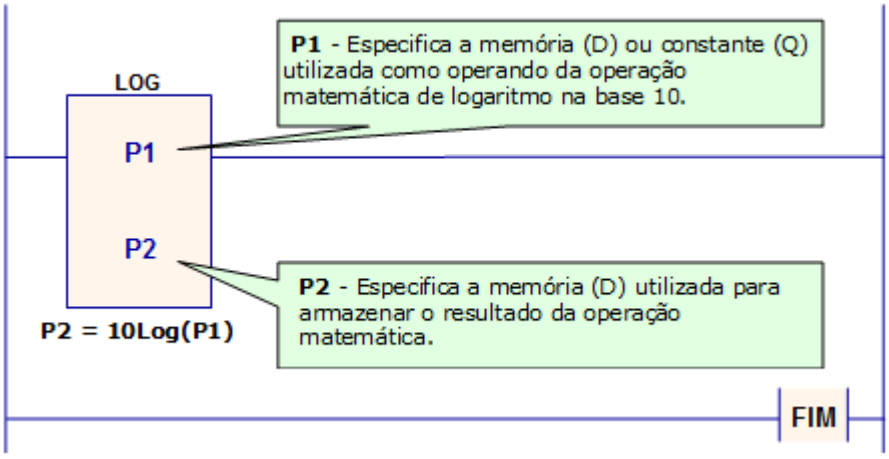


Além de realizar as operações básicas o SPDS-W também disponibiliza outros blocos que são:

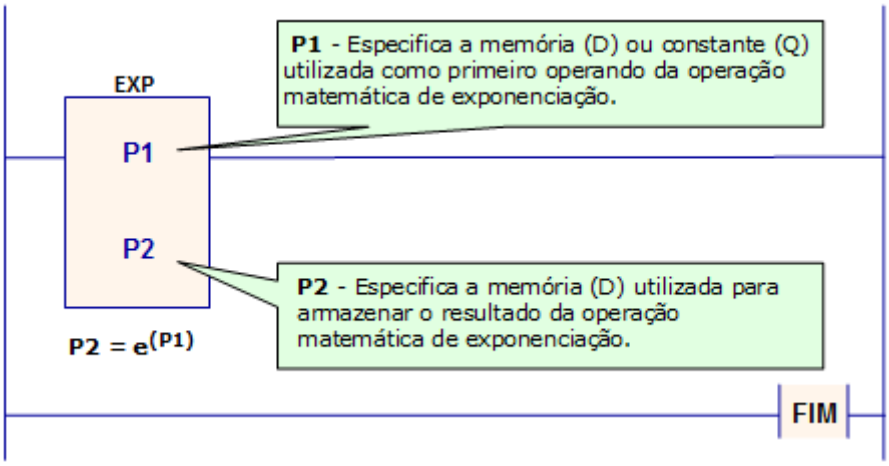
Extrator de raiz quadrada



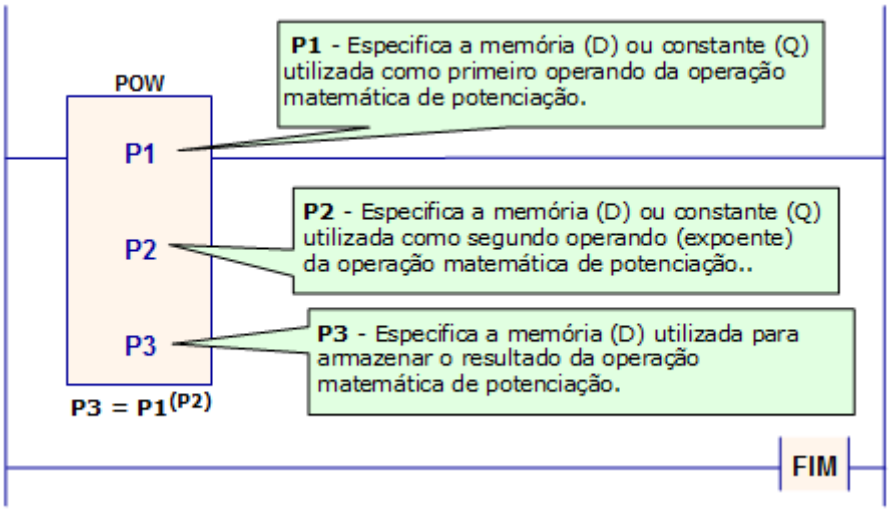
Log na base de 10



Exponenciação



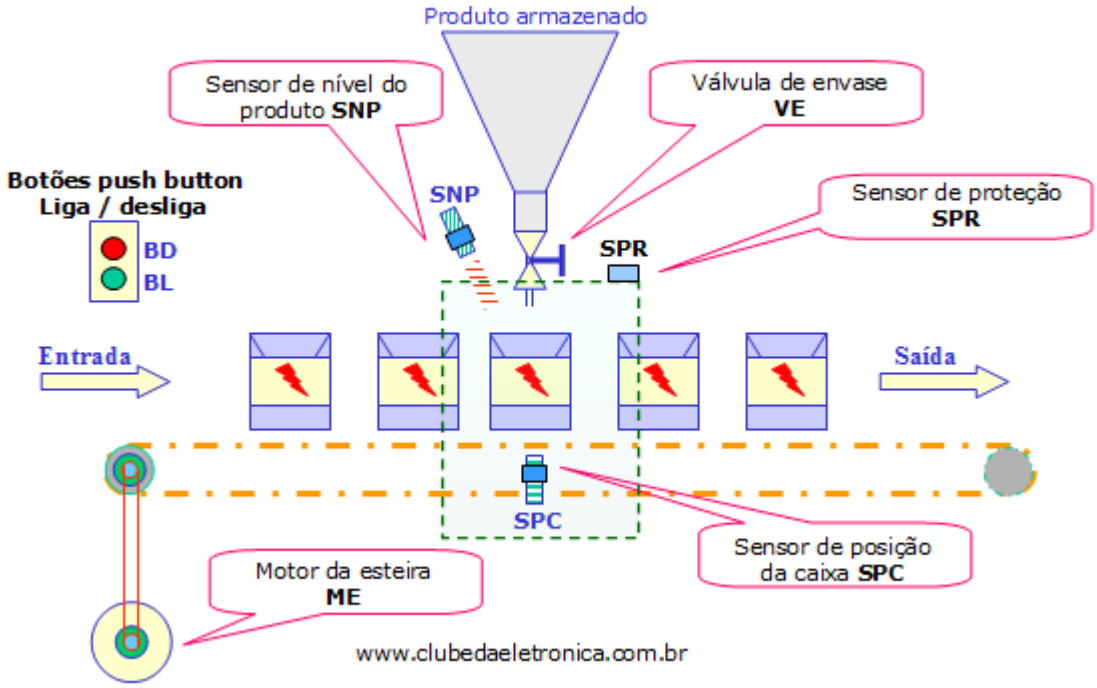
Potenciação



Praticando...

1- Sabendo-se que o timer possui uma base de tempo de 10ms. Elabore um programa capaz de converter o valor apresentado na memória do timer em segundos, e o apresente no display.

3- Deseja-se implementar um sistema para envase de produtos, conforme ilustrado na figura abaixo:



Descrição de funcionamento:

Ao pressionar o botão de Liga **(BL)**, dará início ao processo e a esteira **(ME)** será ligada e só pára quando a caixa chegar à posição de envase, dado pelo sensor de posição da caixa **(SPC)**, neste momento, abre-se a válvula de envase **(VE)** liberando o produto que terá seu nível controlado pelo sensor (**SNP)** e assim que este nível for alcançado liga-se novamente a esteira até que uma nova caixa chegue a posição de envase.

Condição de partida: O sistema só ligará se a tampa de proteção estiver fechada, ou seja, se o sensor de proteção **(SPR)** estiver acionado.

Pressionando o botão de desliga **(BD)** o ciclo será interrompido.

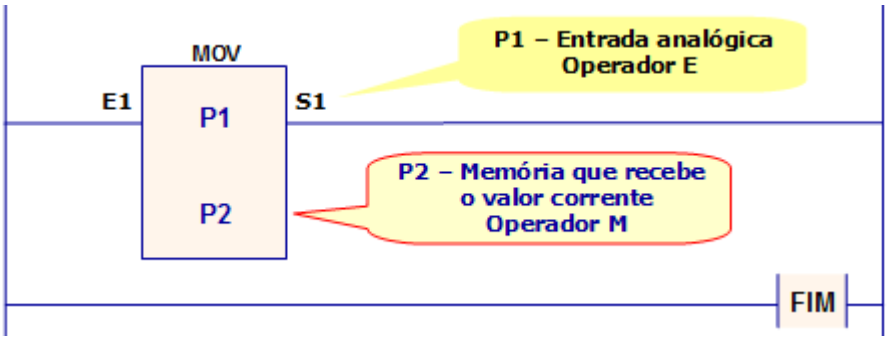
1- Defina as entradas e saídas (I/O) relacionando a sigla com o contato do CLP.
2- Fazer o diagrama de estado da solução proposta.
3- Fazer programa ladder da solução proposta.

Trabalhando com variáveis analógicas

Os Controladores Lógicos Programáveis são equipamentos digitais, ou seja, só entendem “0” e “1”, porém a grande maioria deles possui internamente conversores AD (entrada de sinais) e DA (saída de sinais) na maioria dos casos estes conversores são de 10 bits.

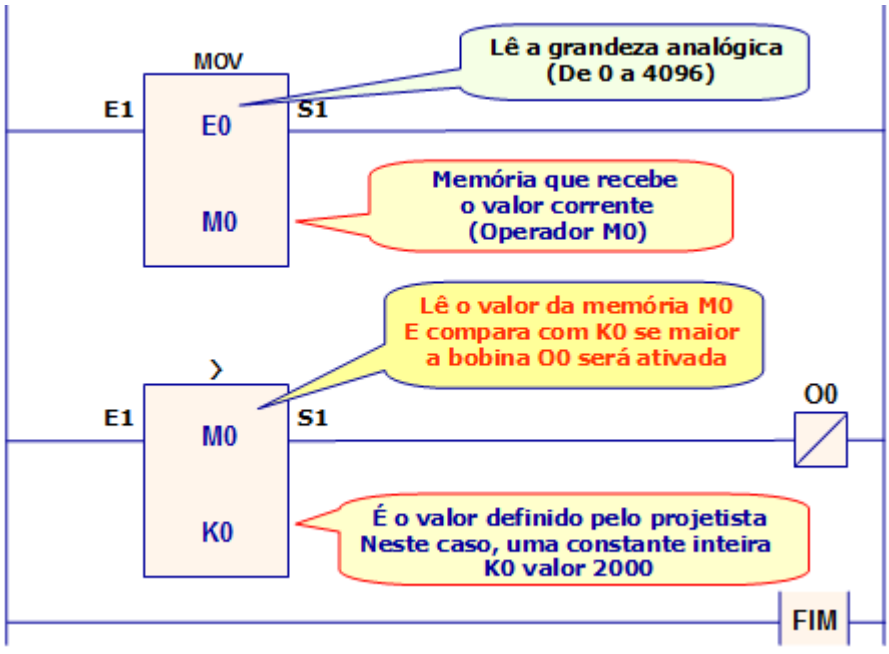
Os controladores lógicos são preparados para receber ou enviar sinais em tensão (tipicamente 0 a 10V) ou em corrente (tipicamente 4 a 20mA), cabendo ao usuário uma consulta ao manual do fabricante.

Segue um exemplo onde o valor corrente do operador E0 será transferido para a memória M0.



Blocos comparadores (=, ⟨⟩, ⟨, ⟨=, ⟩ , ⟩= e &)

Comparar duas variáveis e tomar decisão é sem dúvida de extrema importância para automação. Os CLPs disponibilizam vários blocos para este fim. Vejamos alguns exemplos:



Outros blocos de comparação

P1 Igual a P2 (P1 = P2)

O objetivo destes elementos é realizar a comparação entre operadores. Esta comparação é do tipo que verifica se o operando P1 é igual ao operando P2 Para inseri-lo no programa deve-se posicionar o cursor na posição desejada, selecionar no menu à esquerda o grupo "Comparação" e clicar no botão correspondente.

P1 diferente de P2 (P1 <> P2)

O objetivo destes elementos é realizar a comparação entre operadores. Esta comparação é do tipo que verifica se o operando P1 é diferente do operando P2. Para inseri-lo no programa deve-se posicionar o cursor na posição desejada, selecionar no menu à esquerda o grupo "Comparação" e clicar no botão correspondente.

P1 maior que P2 (P1 > P2)

O objetivo destes elementos é realizar a comparação entre operadores. Esta comparação é do tipo que verifica se o operando P1 é maior que o operando P2. Para inseri-lo no programa deve-se posicionar o cursor na posição desejada, selecionar no menu à esquerda o grupo "Comparação" e clicar no botão correspondente.

P1 maior ou igual a P2 (P1 >= P2)

O objetivo destes elementos é realizar a comparação entre operadores. Esta comparação é do tipo que verifica se o operando P1 é maior ou igual ao operando P2. Para inseri-lo no programa deve-se posicionar o cursor na posição desejada, selecionar no menu à esquerda o grupo "Comparação" e clicar no botão correspondente.

P1 menor que P2 (P1 < P2)

O objetivo destes elementos é realizar a comparação entre operadores. Esta comparação é do tipo que verifica se o operando P1 é menor que o operando P2. Para inseri-lo no programa deve-se posicionar o cursor na posição desejada, selecionar no menu à esquerda o grupo "Comparação" e clicar no botão correspondente.

P1 menor ou igual a P2 (P1 <=P2)

O objetivo destes elementos é realizar a comparação entre operadores. Esta comparação é do tipo que verifica se o operando P1 é menor ou igual ao operando P2. Para inseri-lo no programa deve-se posicionar o cursor na posição desejada, selecionar no menu à esquerda o grupo "Comparação" e clicar no botão correspondente.

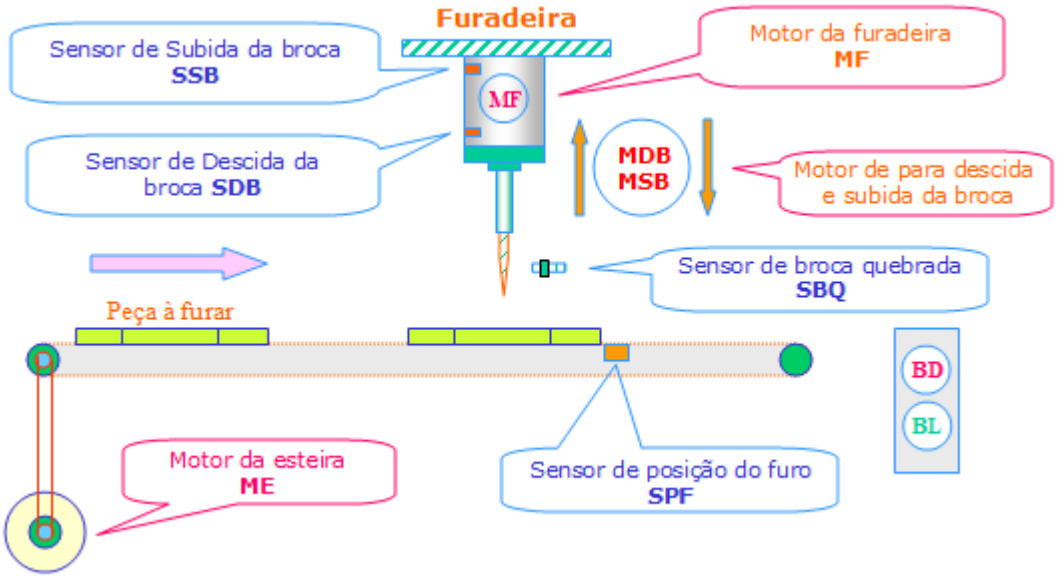
Teste lógico (P1 & P2)

O objetivo deste elemento é realizar a operação lógica AND (E) bit a bit entre dois operadores. Para inseri-lo no programa deve-se posicionar o cursor na posição desejada, selecionar no menu à esquerda o grupo "Comparação" e clicar no botão correspondente.

Praticando...

1- Elabore um programa capaz de:
   a. Ler um sinal analógico de 0 (0°C) a 5V (100°C).
   b. Apresentar o valor da temperatura no display.
   c. O circuito deverá acionar ligar uma ventoinha quando o set point (40°C) for alcançado
   d. A ventoinha deverá ficar ligada por 5 segundos.
   e. Implemente um set point ajustável via IHM

2- Deseja-se programar um sistema para furação de peças, conforme ilustrado na figura abaixo:



Descrição de funcionamento:

Ao pressionar o botão de Liga **(BL)**, dará inicio ao processo e a esteira **(ME)** será ligada e só para quando a peça chegar à posição de furo, dada pelo sensor de furo **(SPF)**, neste momento, o motor de furo **(MF)** e o motor de descida da broca **(MDB)** serão acionados até que o furo esteja completo, dado pelo sensor de descida da broca **(SDB)**, neste momento, é acionado o motor de subida da broca **(MSB)**. Deve-se manter o motor de furo acionado para que a broca não quebre. Quando o sensor de subida da broca **(SSB)** for alcançado o sistema recomeça.

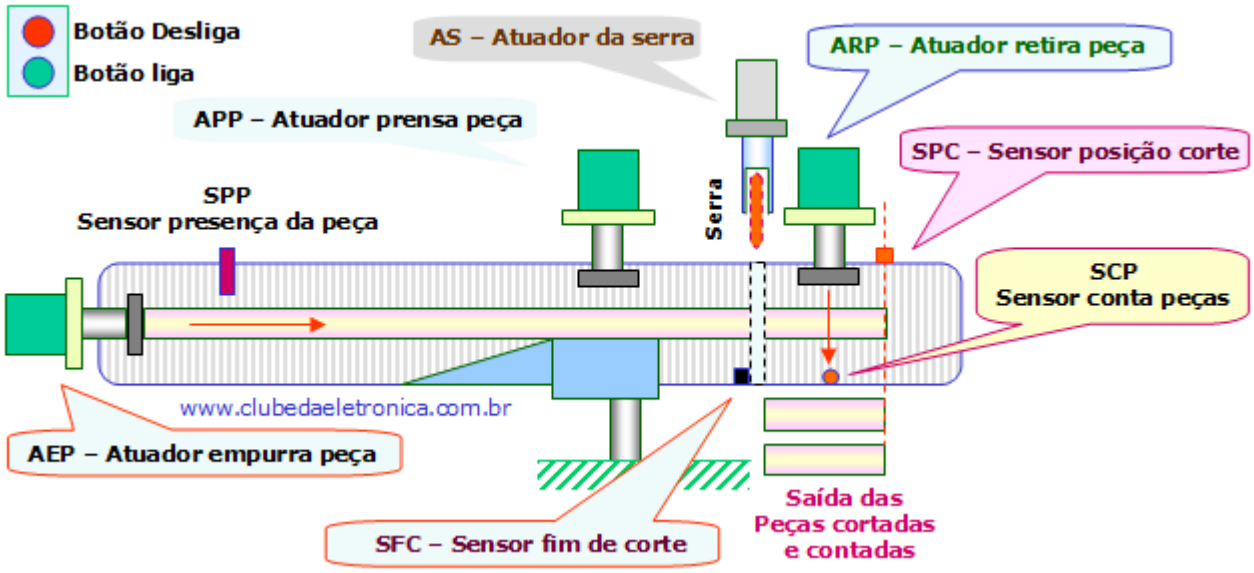
Condições de partida:

O processo só terá início se a broca estiver na posição alta, assim se estiver baixa ela deverá subir. Se estiver quebrada fica esperando a troca e o sistema não liga.

1- Defina as entradas e saídas (I/O) relacionando a sigla com o contato do CLP.
2- Fazer o diagrama de estado da solução proposta.
3- Fazer programa ladder da solução proposta (utilizar comentários)

Projeto máquina de corte automatizada

Deseja-se projetar uma serra automatizada, vejamos os critérios do cliente.



Descrição de funcionamento

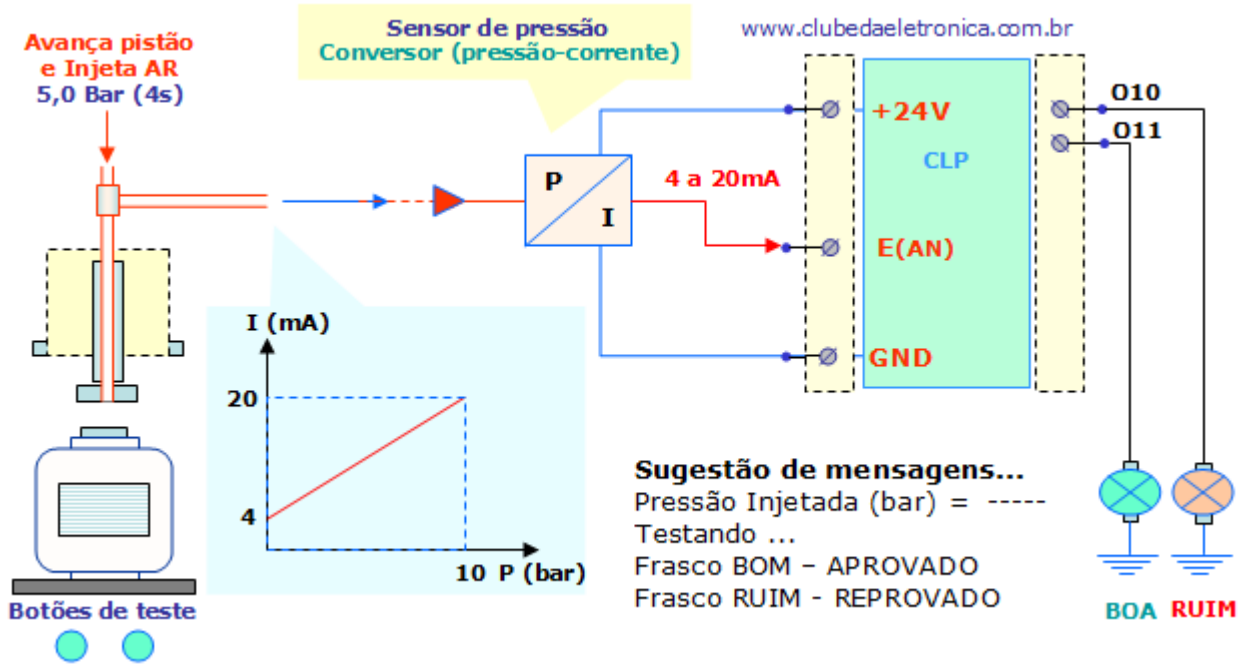
Coloca-se manualmente uma peça de 10 metros na máquina, um sensor detecta a presença da peça **(SPP)** se este for "1" e o botão liga **(BL)** for pressionado, ou seja, "1" a máquina liga, caso contrário, acenderá uma lâmpada **(L)** e na IHM deverá ser exibida a seguinte mensagem (COLOQUE A PEÇA). Uma vez **BL** e **SPP** = 1, liga o atuador de empurra peça **(AEP)** até que a peça encontre o sensor de posição de corte **(SPC)**, neste momento, desliga o **(AEP)** e liga o atuador prensa peça **(APP),** após 1 segundo liga a **SERRA** e o atuador da serra **(AS)** que ficarão acionados até que o sensor de fim de corte **(SFC)** seja alcançado e então aciona o atuador de retirada da peça **(ARP)** e o atuador que empurra a peça para corte **(AEP)**. Quando a décima peça passar pelo sensor **(SCP)** a lâmpada **(L)** acenderá e a mensagem **(COLOQUE A PEÇA)** aparecerá novamente.

Etapas para elaboração do projeto

- Definir o problema corrigindo possíveis falhas no processo.
- Identificar as variáveis e criar a lista de entradas e saídas.
- Montar o diagrama de estado da solução proposta.
- Implementar no Kit do Zap500 (não esquecer dos comentários e identificação das variáveis)

Teste de estanqueidade

Coloca-se o frasco manualmente na posição de teste, pressiona-se os botões de teste simultaneamente (push button), o cilindro avança, depois de 1 segundo injeta-se ar (5 bar) por 4s, desliga-se o ar, o sensor mede a pressão por um tempo de 5s), e se neste tempo a pressão cair o franco será rejeitado (exibir mensagem no display: **FRASCO RUIM**), se estiver não cair neste intervalo de tempo o frasco esta bom (exibir mensagem no display: **FRASCO BOM**).



"Toda idéia brilhante de hoje já foi uma idéia impraticável no passado."

Bill Gates.

www.clubedaeletronica.com.br

Referências bibliográficas:

- Circuitos digitais, Autor: Antonio Carlos de Oliveira Lourenço, Ed. Érica.
- http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/
- http://www.cpdee.ufmg.br/~carmela/NORMA%20IEC%201131.doc
- http://www.software.rockwell.com/corporate/reference/Iec1131/
- http://www.plcopen.org/
- http://www.lme.usp.br/~fonseca/psi2562%20aula%206%20IHM.pdf
- http://www.teses.usp.br/teses/disponiveis/18/18133/tde-11072002-085859/
- http://www.redenet.edu.br/publicacoes/arquivos/20080108_144615_INDU-058.pdf
- http://www.corradi.junior.nom.br/modCLP.pdf
- http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/