Laboratório de automação industrial

Teoria geral de automação industrial

Controladores Lógicos Programáveis

CAP – Controladores de Automação Programável

PAC – Programmable Automation Controller

Prof. Dr. Sergio Luiz Pereira

2012

Índice

1 Teoria geral da automação

Desde o início da primeira Revolução Industrial a engenharia teve que lidar com processos executados, supervisionados e controlados pelo homem. Simultaneamente começaram os esforços no desenvolvimento da teoria, da tecnologia e da ciência de automação no intuito de substituir totalmente ou pelo menos parcialmente a atividade humana no controle, na supervisão e na execução dos mesmos processos.

Mais de um século depois, a ciência de controle e de automação tornou-se mais do que nunca necessária e está presente não somente nos setores produtivos como também nos setores de serviços.

O controle de processo pode ser: manual, semi-manual ou automático. As fronteiras entre um e outro nem sempre são bem delineadas, porém o conceito é importante. Num controle de processo 100% manual todas as ações são de total responsabilidade e de dependência humana. Num controle de processo 100 % automático todas as ações são efetuadas sem a interferência humana. Um exemplo seria dois tipos de processos de soldagem, onde o primeiro é efetuado de forma automática, por meio de robôs, e o outro efetuado de forma manual, por meio de operadores

A definição clássica de Automação da Produção estabelece três tipos de classificação envolvendo a aplicação de dispositivos e equipamentos mecânicos, eletrônicos, computacionais e de telecomunicações, Os três tipos são:

a) Automação fixa;

b) Automação programável, e;

c) Automação flexível.

A figura 1 apresenta a curva genérica de aplicação de utilização dos três tipos de automação, conforme o nível de produção.

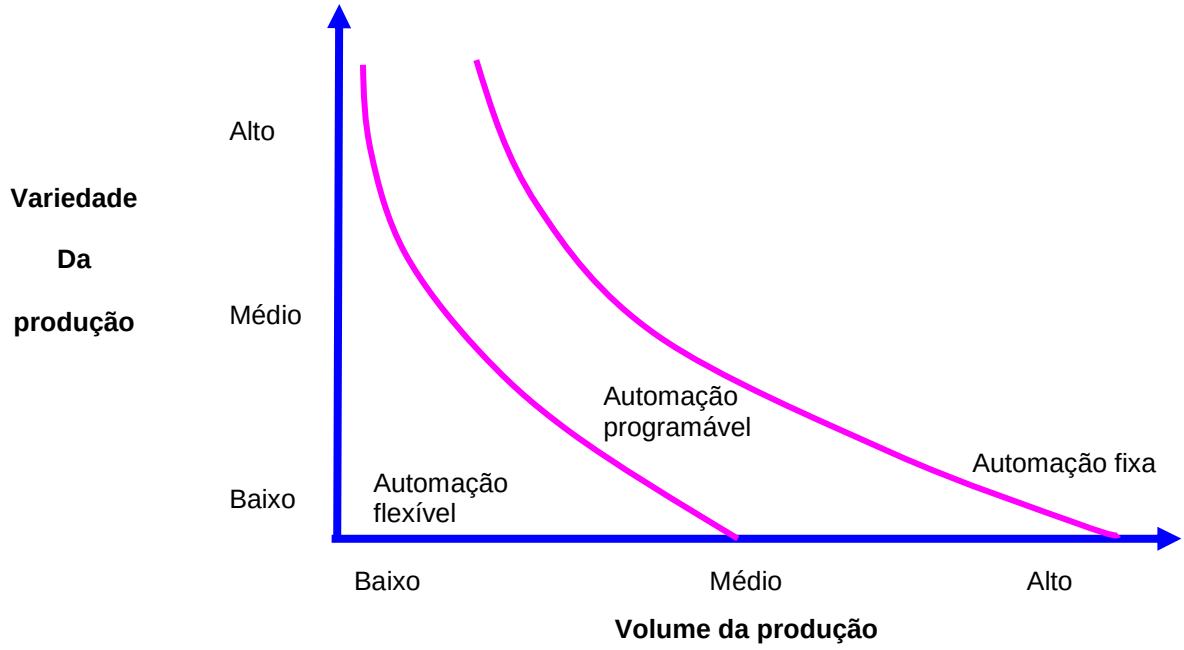


Figura 1 - Tipos de automação conforme o volume de produção e a variedade da produção.

2 Teoria geral de Controladores Lógicos Programáveis

2.1 Circuitos de intertravamento com reles

Relés, relés temporizados e contatores são dispositivos eletromecânicos que permitem a implementação por meio elétrico de algoritmos baseados em lógica booleana. Assim sendo, os mesmos também podem ser empregados, em diversas circunstâncias, para implementar a automação de determinados sistemas a eventos discretos.

As primeiras linhas de manufatura automatizadas, assim como os primeiros elevadores automáticos empregavam exclusivamente relés, relés temporizados e contatores para implementarem os processos ou as lógicas de controle definidas para os determinados conjuntos de situações e de eventos discretos.

Normalmente os relés, os relés temporizados e os contatores ficam instalados em um ou mais painéis específicos e são interligados por meio da fiação elétrica aos diversos dispositivos de campo como sensores, chaves de fim de curso, botoeiras, sinalizadores, motores etc.

A figura 2 ilustra um exemplo genérico e simplificado de um painel de comando constituído de relés, contatores e de botoeiras que estão ligados a diversos dispositivos de campo. O painel representado na figura 2 possui um circuito interno conectado aos dispositivos de campo de sensoriamento e de acionamento. A representação das botoeiras normalmente fechadas ou normalmente abertas, os contatos normalmente abertos ou normalmente fechados dos relés, e as suas bobinas, seguem um padrão estipulado e definido como diagrama contatos ou de relés.

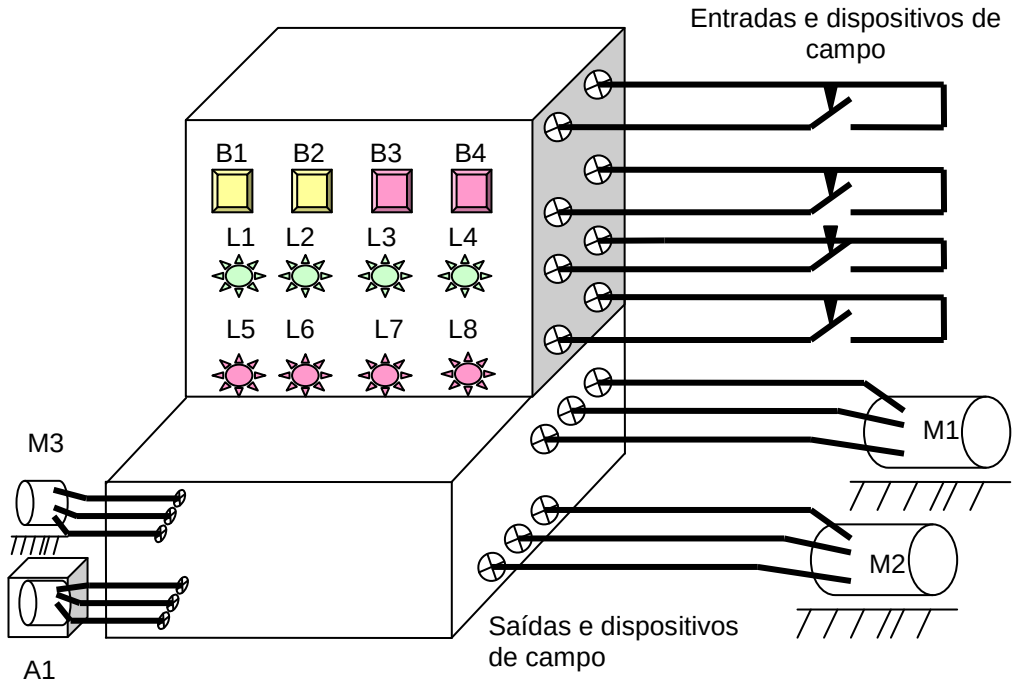


Figura 2 - Ilustrativo de um painel de relés e parte do circuito elétrico.

No exemplo do circuito mostrado na figura 3 percebe-se que, caso a botoeira B1 seja acionada, a bobina do relé R1 será energizada. Se tal fato ocorrer o contato normalmente aberto do relé R1 (contato 10) será fechado e, por conseqüência, dará continuidade elétrica ao seu respectivo circuito elétrico. Também o contato normalmente fechado (contato 11) será aberto e por conseqüência provocará a interrupção da corrente elétrica do seu respectivo circuito. Entretanto, os contatores C1 e C2 somente serão energizados caso os dispositivos sensores de campo que estão ligados aos terminais C1a/C1b e aos terminais C2a/C2b estiverem em uma condição que também permita a continuidade da corrente elétrica.

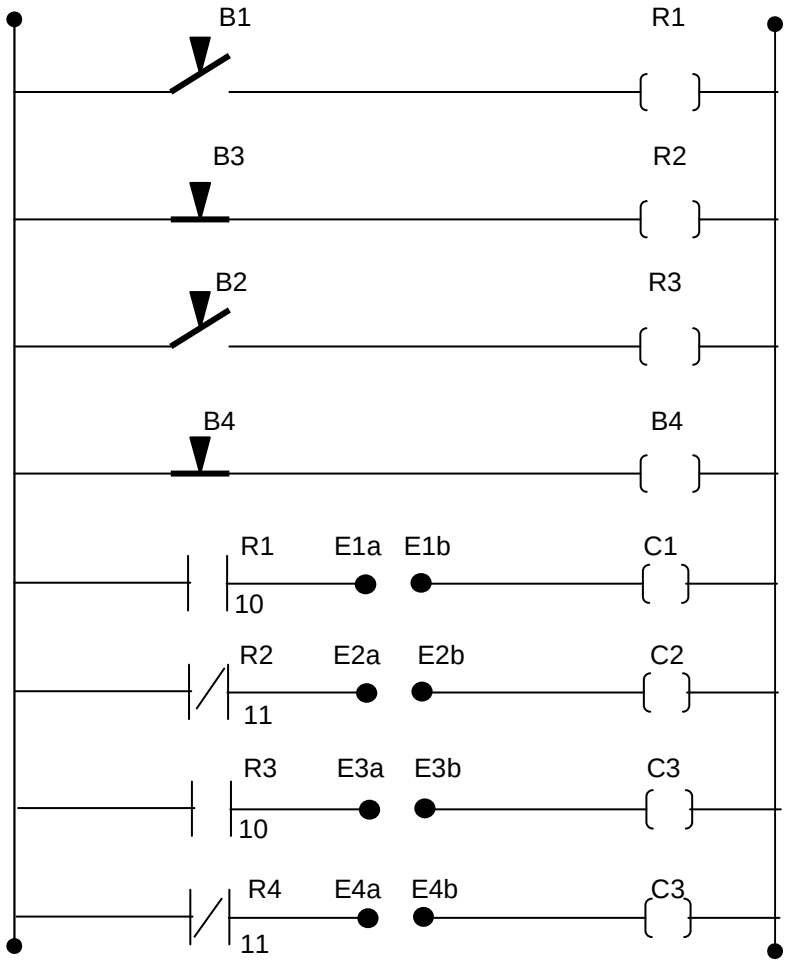
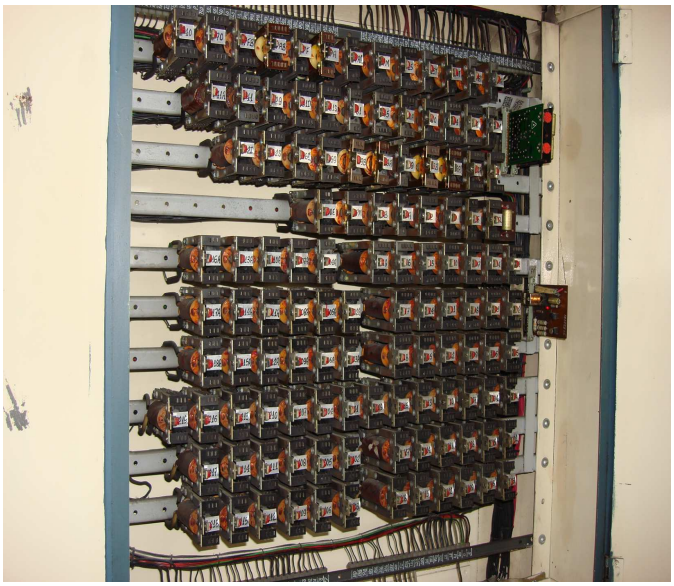


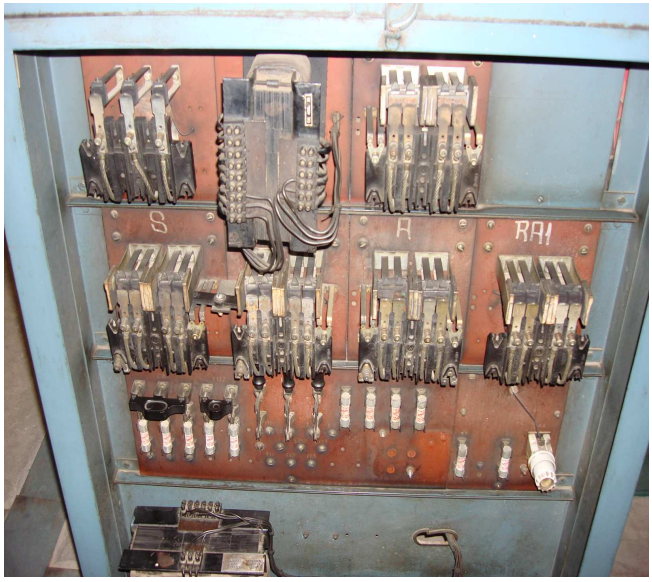
Figura 3 - Ilustrativo parcial de um circuito elétrico com relés e contadores.

Um painel de comando a relé pode incorporar ainda diversos dispositivos como temporizadores, sinalizadores, dispositivos de proteção etc. Um painel de comando a rele pode ser empregado para automatizar diversos tipos de processos, porém, o algoritmo do processo implementado está intimamente conectado com o hardware. Ou seja, caso haja a necessidade de alguma alteração no intertravamento lógico de monitoração e de controle do processo, o circuito elétrico também eventualmente também os dispositivos deverão ser alterados.

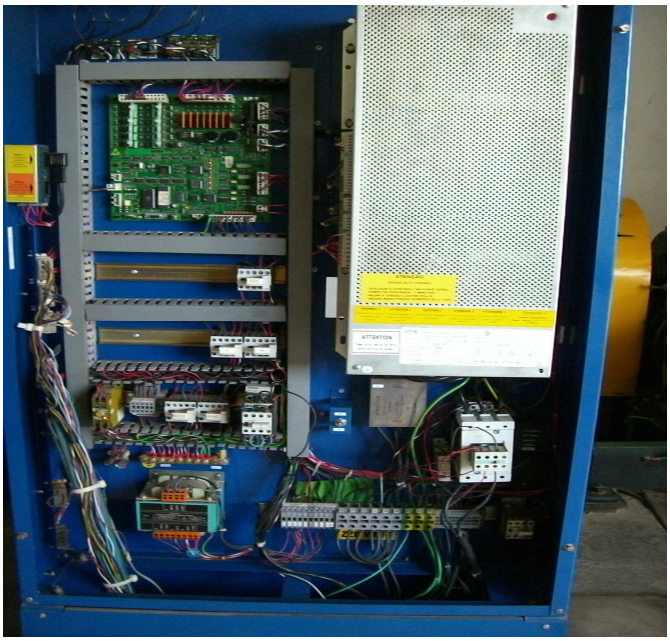
As figuras 4.a e 4.b ilustram duas fotografias de um painel de comando e controle eletromecânico, locado na casa de máquinas dos elevadores de um edifico construído nos anos 80. As figuras 5.a e 5.b ilustram duas fotografias do painel de comando (tecnologia atual) que substitui o painel antigo. Observa-se que o mesmo continua

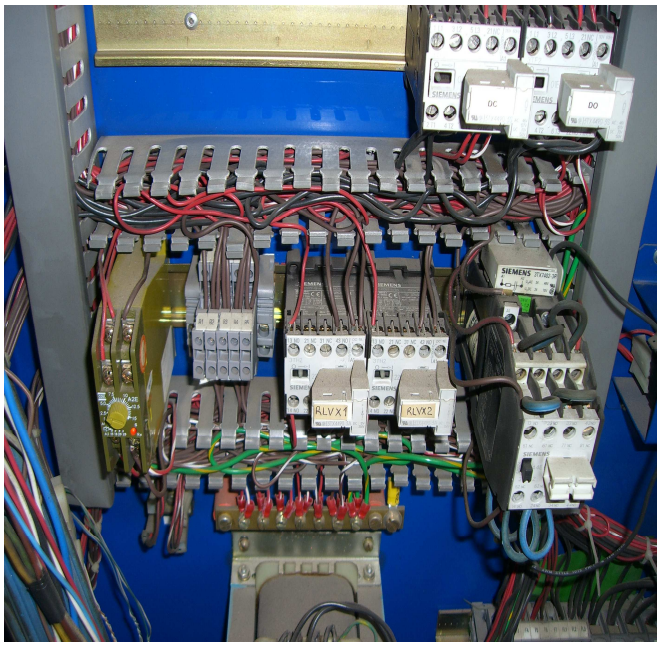
fazendo uso de contatores para o acionamento de potência. Entretanto, a partida dos motores neste sistema não é mais por acionamento com partida do motor ligado em estrela, para depois reverter as ligações para triângulo, mas sim com acionamento fazendo emprego de um inversor. Observa-se também que parte deste painel emprega dispositivos de estado sólido para realizar a lógica de intertravamento do algoritmo de controle do elevador.





Figuras 4a e 4b - Fotografias de um painel de comando eletromecânico para elevadores.





Figuras 5a e 5b - Fotografias de um painel de comando eletromecânico e também micro processado que emprega inversor de freqüência para o acionamento do motor do elevador.

2.2 Histórico e arquitetura de hardware dos Controladores Programáveis

O desenvolvimento do CLPs (Controladores Lógicos Programáveis) ou CPs (Controladores Programáveis) começou por volta de 1968 quando a General Motors solicitou à indústria eletrônica uma alternativa tecnológica de um equipamento de estado sólido e programável para a implementação da automação até então implementada em painéis de comando repletos de dispositivos eletromecânicos.

As principais razões que motivaram essas solicitações foram:

a) O sistema de relé utilizado na automação das linhas de manufatura, de montagem, de carregamento e de controle de máquinas havia se tornado grande e complexo.

b) Necessidade do melhorar os índices de confiabilidade e de diminuir os custos de manutenção dos sistemas de automação.

c) Necessidade de se ter flexibilidade e agilidade nas implementações e nas alterações das plantas automatizadas e nos seus respectivos sistemas de automação.

Os Controladores Lógicos Programáveis são equipamentos de estado sólido projetados inicialmente para substituir as ações dos relés e temporizadores num circuito de intertravamento como o mostrado no exemplo anterior. Entretanto, os CLPs atuais incorporam funções avançadas como: controle estatístico, controle de malha, comunicação em rede, etc.

Os CLPs são projetados e construídos para operarem em ambientes severos e, portanto, devem resistir a altas temperaturas, ruídos elétricos, poluição atmosférica, umidade, etc. Sua capacidade quanto ao número de entradas e saídas, memória, conjunto de instruções, velocidade de processamento, conectividade, flexibilidade, IHM (interface homem máquina) e etc. varia conforme o fabricante e o modelo.

Existe no mercado uma grande variedade de modelos de grande, médio e pequeno porte. A sua arquitetura básica e genérica é composta de: Unidade Central de Processamento (UPC), Memória do Usuário onde é armazenado o programa de aplicação do usuário, Cartões de entrada e de saída (E/S) também chamados de Cartões de I/O,

Fonte de alimentação. Os CLPs também possuem interface serial para dispositivos de Programação e de Leitura (IHM), módulos de rede, etc. A figura 6 ilustra o diagrama de blocos de um CLP genérico.

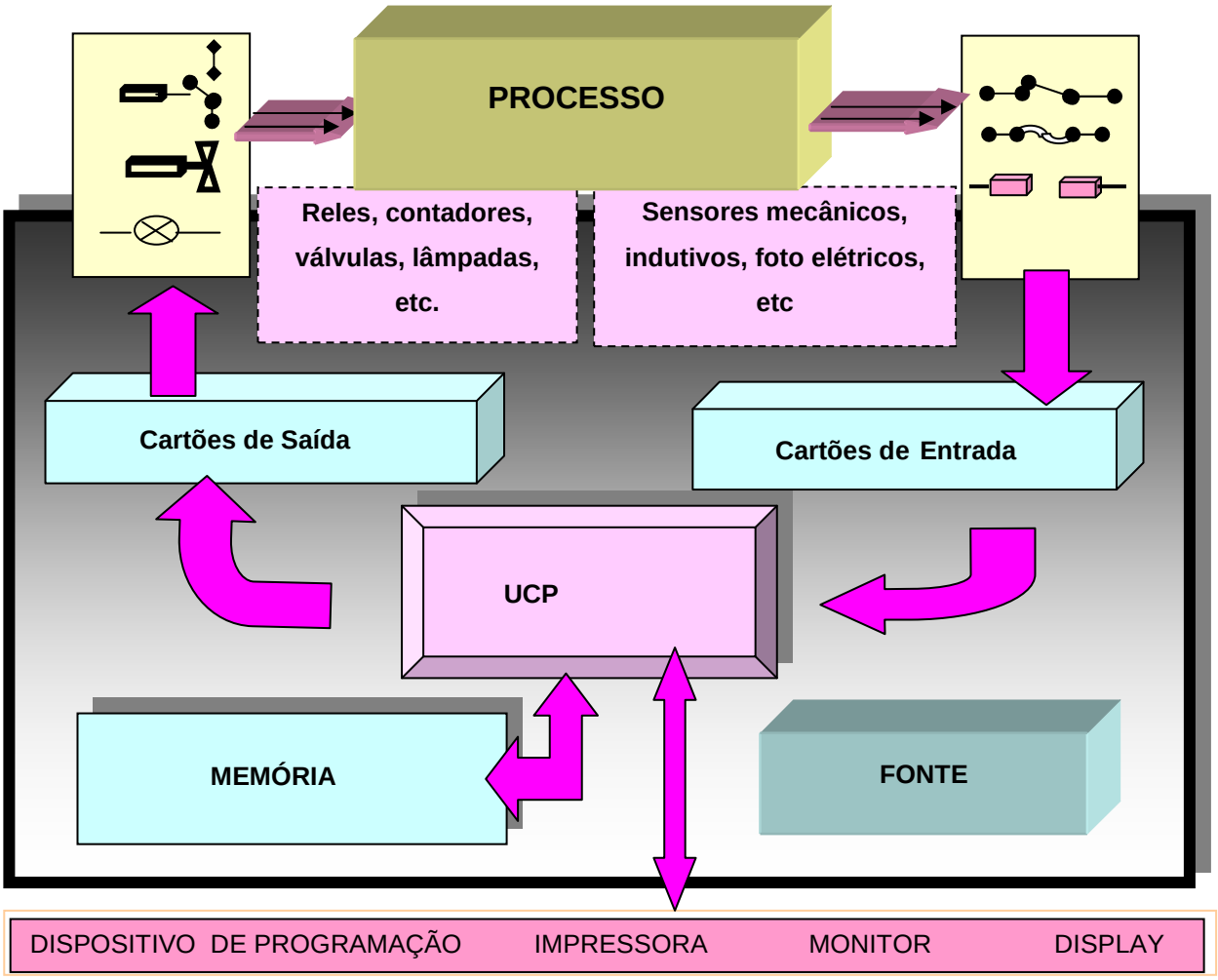


Figura 6 Ilustrativo da arquitetura de hardware de um CLP genérico.

Deve-se ressaltar que os CLPs tiveram uma grande evolução na sua capacidade de programação e de execução de tarefas. No início dos anos 2000 surgiu uma nova geração de CLP denominada da PAC ou CAP (Controlador de Automação Programável)

Um CAP possui tanto a elevada confiabilidade de hardware dos CLPs, quanto a elevada capacidade de processamento e quantidade de memória dos microcomputadores. A principal proposta de um CAP é de ter as funções de um SDCD (sistema Discreto de Controle Distribuído) com o preço e a simplicidade do CLP, que assim como os CLPs, opere em conjunto com sistemas supervisórios ou SCADA (Supervisory Control And Data Acquisition) de forma que os principais benefícios e

funções dos dois tipos de sistemas de automação SDCD e CLP conjuntamente com SCADA possam ser unificados em um único tipo de sistema. Os CAP permitem rodar programas complexos, elaborados em linguagem de alto nível, conforme as necessidades do sistema de automação. Os PAC's são controladores com capacidade de lidar com controles envolvendo lógica, movimento "motion" e processo num mesmo controlador.

2.3 Arquitetura de hardware

Alguns dos componentes principais de um CLP tradicional, como mostrados na Fig. 6 (fonte, módulo de comunicação, entrada digital, saída digital, entrada analógica, saída analógica e UCP) são explicados com mais detalhes a seguir.

2.3.1 Fonte

Alimentada em CA ou CC fornece os níveis de tensão necessários à operação da CPU e das interfaces. Muitas vezes, oferece fonte auxiliar de 24VCC destinada à alimentação de transdutores, relés, módulos de interface, etc.

2.3.2 Módulo de comunicação

Permite que o CLP se comunique com outros equipamentos, tais como outros CLPs, um computador encarregado de supervisionar o processo, ou com uma rede de sensores e atuadores de campo. O meio de comunicação pode ser uma rede de comunicações serial, nos padrões RS-232, RS-422 ou RS-485, ou uma moderna rede Ethernet, usando meios como cabos seriais ou de rede, em cobre ou fibra óptica. É importante ilustrar que uma rede de comunicação, com um simples par de fios ligando um CLP a uma rede de sensores e atuadores, é capaz de captar as informações do processo e enviar as decisões de comando aos atuadores, proporcionando uma substancial simplificação à cablagem de um sistema.

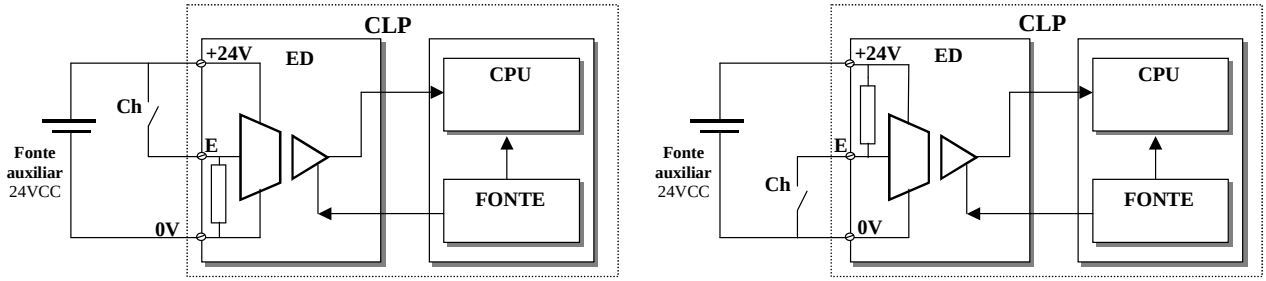
A utilização de tais redes na automação também permite que se obtenham, em tempo real, informações de todo o processo produtivo, incluindo estocagem de matéria prima, linha de produção, estocagem de produto acabado, etc.

2.3.3 Entrada Digital (ED)

Permite que sinais binários, do tipo “falso-verdadeiro” sejam monitorados e armazenados pelo CLP. Alguns exemplos de dispositivos usualmente conectados às EDs são: botoeiras, chaves fim de curso, contatos de relés, pressostatos, termostatos, relés de proteção, cortinas de luz, sensores de presença, sensores de proximidade, encoders, etc.

As entradas digitais dividem-se em duas categorias, as de corrente continua, com nível de tensão de 24V, e as de corrente alternada com opções de 110V ou 220V, além de algumas outras configurações particulares. As opções mais comuns são mostradas a seguir.

a) Entradas do tipo CC: As entradas CC são apresentadas na configuração positiva (sink) e negativa (source), conforme ilustrado nas figuras 7a e 7b.



7a. ED positiva (sink) 7b. ED negativa (source)

Figuras 7a e 7b - Entradas digitais do tipo CC (Fonte: MATAKAS, L. PEREIRA)

Nas duas configurações das figuras 7a e 7b, o sinal de 0/24VCC na entrada digital é aplicado a um amplificador isolador, responsável pelo isolamento galvânico elétrico entre os ambientes elétricos do processo e da UCP. A isolação usual de tensão é de 2500VCA. Na maioria dos casos a isolação elétrica é provida via acopladores ópticos. A parte do lado direito do amplificador isolador é alimentada pela mesma fonte que alimenta a UCP. O circuito eletrônico do amplificador deve ser alimentado por fonte auxiliar e isolada de 24VCC.

b) Entradas do tipo CA: A figura 8 ilustra um circuito genérico simplificado de uma entrada CA.

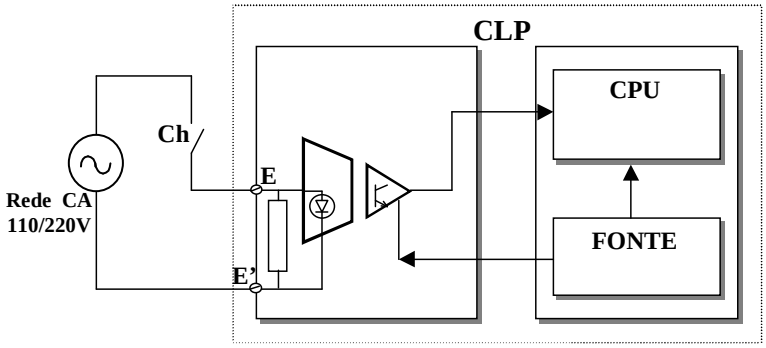


Figura 8 - Entrada Digital do tipo CA (Fonte: MATAKAS, L. PEREIRA).

A tensão alternada (110 ou 220VCA) aplicada aos terminais E e E` ativa o LED (diodo eletro-luminescente), do acoplador óptico que por sua vez envia o sinal à UCP. Normalmente uma entrada digital CA é mais barata que uma entrada digital CC porque a mesma não necessita de fontes auxiliares. Entretanto, o tempo de resposta é maior do que as entradas digitais CC porque são necessários, no mínimo, quase dois ciclos de tensão para que a entrada possa detectar uma real variação do nível de tensão da entrada. (28 mili segundos). Normalmente para aplicações como dispositivos do tipo pressostatos, termostatos, botoeiras, relés térmicos este atraso (*delay*) não é relevante.

Seja uma entrada digital efetuada por um cartão CC ou CA o nível de tensão externo da entrada é convertido em uma variável binária dentro de uma área da memória do CLP. Esta área de memória é organizada conforme o projeto de cada CLP. Esta área também é designada como “imagem da entrada”. Deve-se atentar que os CLPs, dependendo do projeto, podem ter o sistema computacional organizado em palavras “Word” de 8, 16 ou mesmo 32 bits.

A figura 9 ilustra um exemplo genérico de como o estado de uma entrada conectada ao dispositivo de campo é registrado na memória de um CLP.

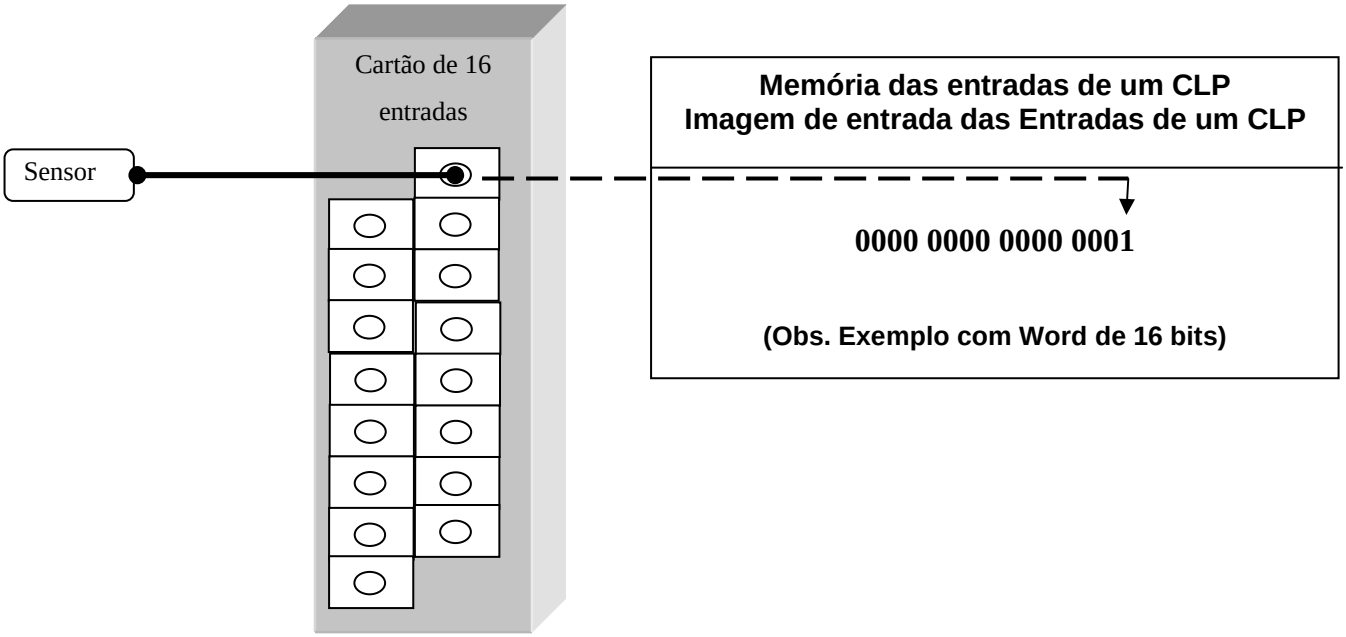


Figura 9 Exemplo de registro do estado de um sensor na memória se um CLP.

A figura 10 ilustra um trecho do manual dos cartões de entrada digital AC da série do CLP SLC500 da Rockwell Automation.

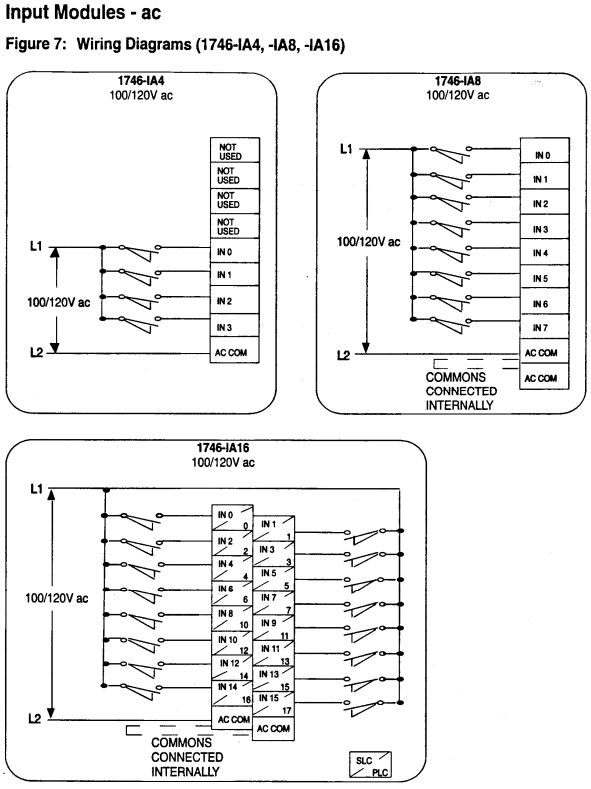
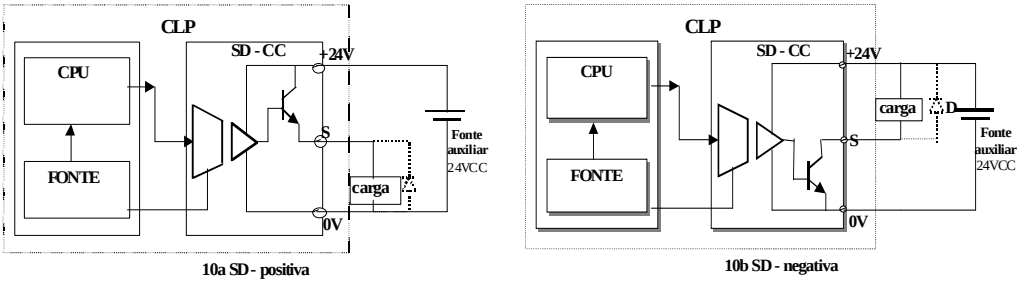


Figura 10 - Ilustrativo de ligação de três tipos de cartões de entrada digital CA.

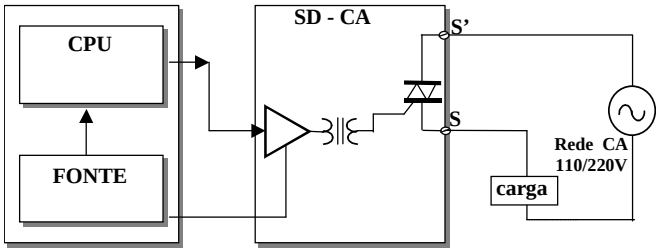
2.3.4 Saídas Digitais

Basicamente são três as categorias de saídas digitais: corrente contínua, corrente alternada e a relé. As figuras 11a e 11b, 12 e 13 ilustram respectivamente: saída digital CC(positiva), saída digital CC (negativa), saída digital em corrente alternada e saída digital à rele.



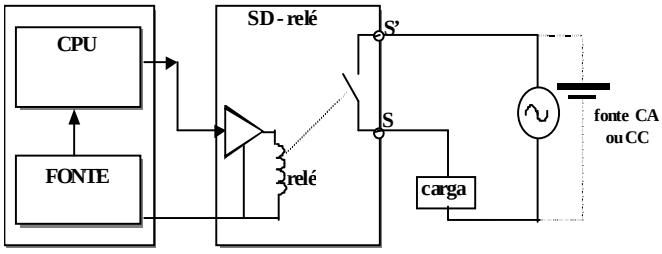
Observa-se que na saída positiva, quando a UCP envia um sinal ao amplificador isolador, o transistor Q, passa a operar no modo de saturação. Assim os terminais "S" e "+24V" são interligados e portanto, a carga fica com uma tensão de 24V. Quando a UCP envia sinal nulo, o transistor opera tal como um circuito em aberto e a carga fica com tensão zero. A saída negativa opera de forma oposta à da saída positiva

Figuras 11a e 11b - Saídas do tipo CC(Fonte: MATAKAS, L. PEREIRA)



A operação é baseada em um TRIAC que recebe os pulsos da UCP via um transformador de isolação. O TRIAC é um dispositivo semicondutor que inicia a condução ao receber um pulso em seu gatilho. Porém, só interrompe a condução quando a corrente se tornar nula.

Figura 12 - Saída digital CA (Fonte: MATAKAS, L. PEREIRA)



Quando a UCP ativa a bobina de um micro-relé localizado dentro do módulo de saída. Um contato é então fechado. Normalmente o contato é dimensionado para comutar cargas em CC ou CA com tensões de até 250V todas de baixa corrente. Observa-se que um cartão com 16 saídas á rele possui 16 reles e 16 contatos que podem ou não estarem ligados em paralelo a um ponto comum. Há a possibilidade também dos contatos serem fornecidos nas configurações NA, ou NF, ou em ambas.

Figura 13 - Saída à rele (Fonte: MATAKAS, L. PEREIRA)

A figura 14 ilustra um exemplo genérico de como um dispositivo de campo, ligado a uma saída digital, é acionado conforme o estado de uma área específica da memória de um CLP.

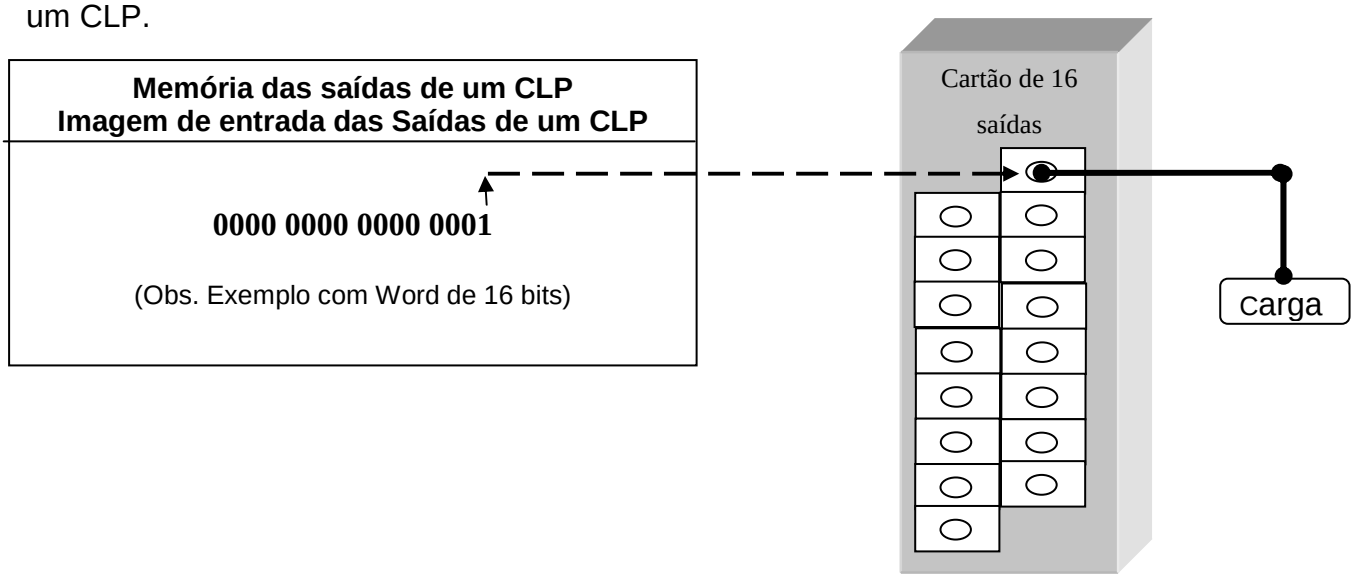


Figura 14 Exemplo de acionamento do estado de uma saída digital em função do estado da memória da área específica imagem das saídas de um CLP.

A figura 15 ilustra um trecho do manual dos cartões de saída digital da série do CLP SLC500 da Rockwell Automation.

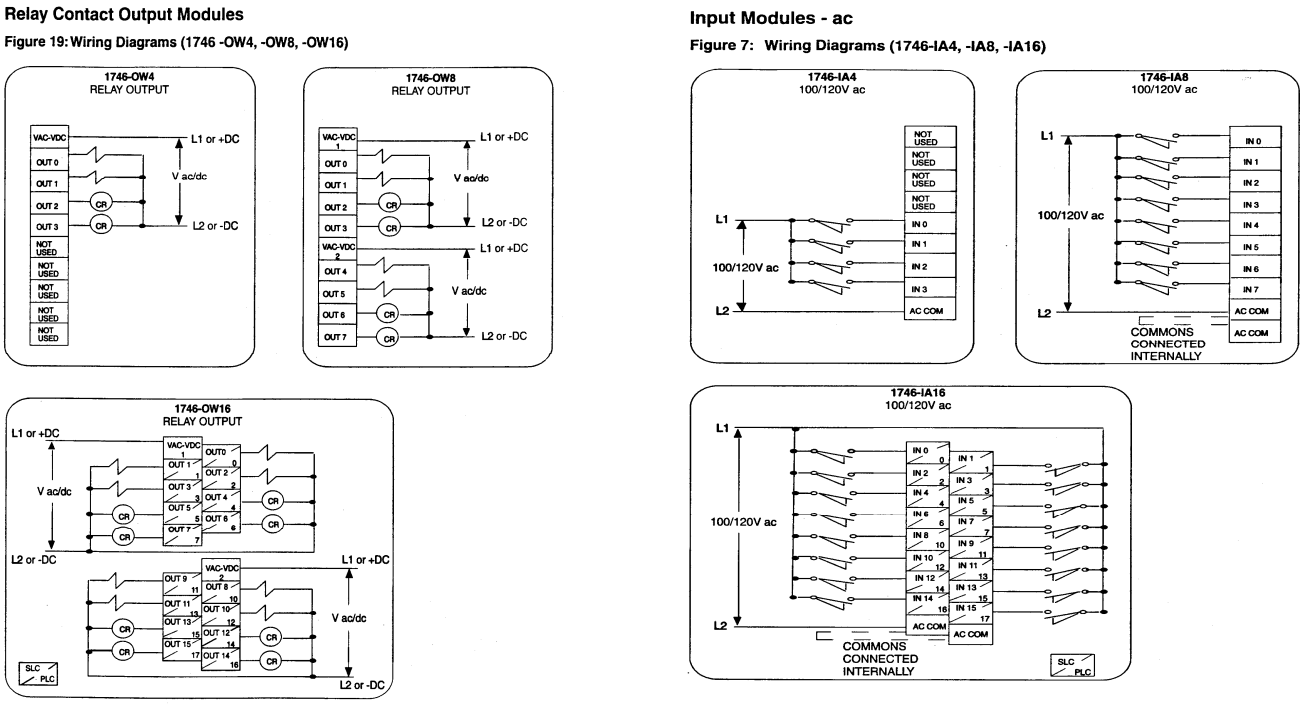


Figura 15 Ilustrativo parcial de ligação de seis tipos de saída digital.

2.3.5 Entradas e Saídas Analógicas

As entradas e saídas analógicas possibilitam que os CLPs também executem um controle de malha por meio de ações de controle como: PID (Proporcional, Integral, Derivativo), controle Fuzzy etc. Permitem ainda que qualquer algoritmo de controle programado no CLPs tome decisões ou ações de controle baseado não somente em valores binários, mas também em valores proporcionais das grandezas do processo controladas como: temperatura, pressão, velocidade etc.

Existem dois tipos de entradas e saídas analógicas: as que lidam com um sinal de tensão proporcional à grandeza medida ou controlada. Normalmente a faixa de tensão é de –10~10V e ou de 0~10V. A faixa de corrente mais empregada é de 4~20mA. A figura 16 ilustra um exemplo genérico de um CLP também executando um controle de malha. A figura 17 reproduz um trecho de manual de um cartão de entrada analógico.

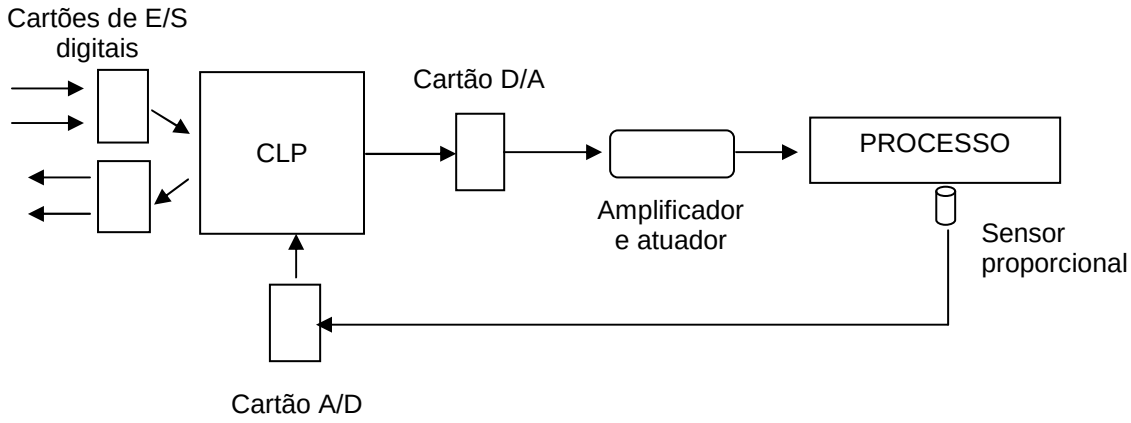


Figura 16 - Ilustrativo de um CLP sendo também empregado em um controle de malha empregando um cartão (AD) ou (EA) e um cartão (DA) ou (SA).

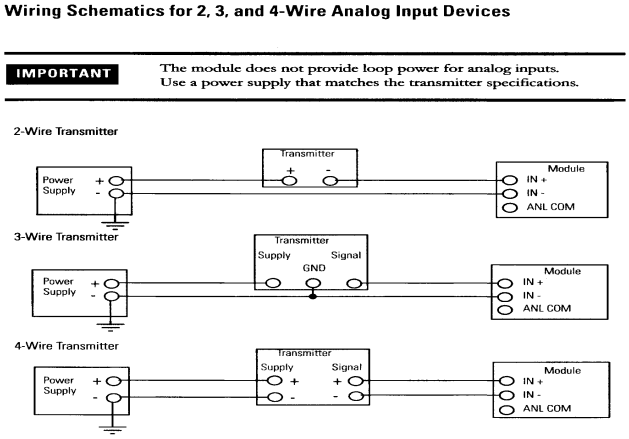


Figura 17 - Ilustrativo parcial de manual de um modelo de entrada analógica.

2.3.6 UCP

A Unidade Central de Processamento é a unidade que executa o programa de controle. Os CLPs tradicionais possuem uma UCP que realiza todas as funções de controle, leitura e escrita na memória. Também existem atualmente CLPs que possuem duas ou mais UPCs operando com divisão de tarefas. A figura 18 ilustra o diagrama de bloco genérico de uma UCP.

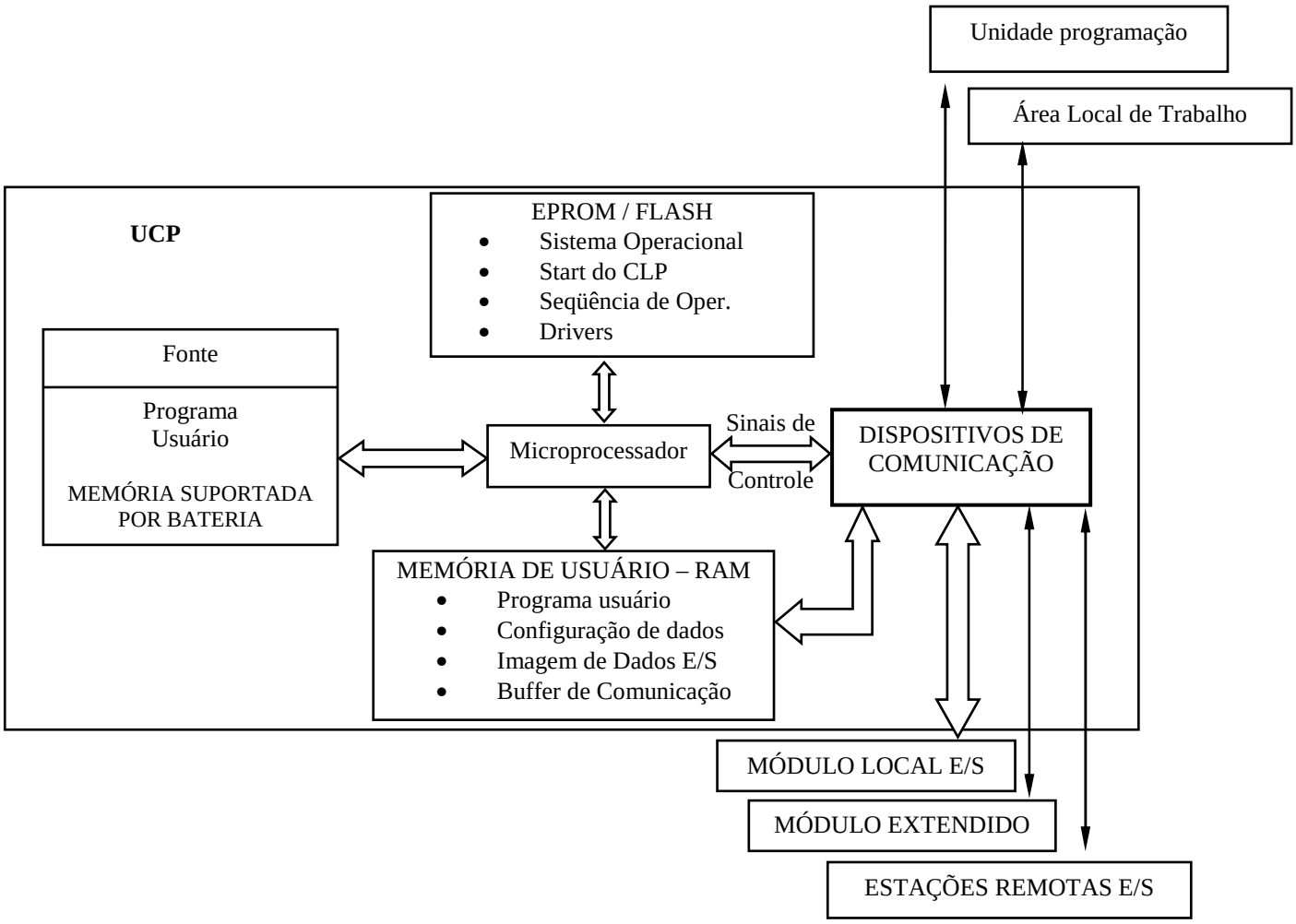


Figura 18 - Diagrama de blocos de uma UCP genérica.

2.4 Controladores Lógicos Programáveis (CLPs) tradicionais

2.4.1 Operação de CLPs tradicionais.

Quando estão na condição de operação “RUN” ou rodando um programa os CLPs operam em uma permanente varredura também denominada SCAN. O SCAN de um CLP é subdividido em três SCANs: SCAN de entrada, SCAN de programa e SCAN de saída. Durante o SCAN de entrada é efetuada a leitura de todas as variáveis e dados disponíveis nos cartões de entrada. Ou seja, é durante o SCAN de entrada que os dados disponíveis nos cartões de entrada são copiados para uma área de memória da RAM geralmente definida como área de imagem de entrada (endereço da memória pré-estabelecido). Terminado o SCAN de entrada a CPU inicia o SCAN de programa. É durante o SCAN de programa que a lógica programada pelo usuário é executada. Terminado o SCAN de programa é iniciado o SCAN de saída quando então os cartões de saída serão atualizados com os dados ou variáveis que estão na área de memória RAM geralmente definida como imagem das saídas de um CLP, (endereço de memória pré-estabelecido).

Diversos modelos de CLPs também apresentam a possibilidade de se alterar o ciclo o SCAN gerando dessa forma quatro tipos básicos de SCAN:

a) Ciclo elementar

   (entradas→programa→saídas).

b) Ciclo de entradas agrupadas, saídas distribuídas)

   (entradas→processamento→saídas→processamento de saídas).

c) Ciclo distribuído

   (E/S ativadas conforme requisitadas) (E→P→O→I→P→.....)

d) Ciclo distribuído com sub “clock” .

A figura 19 ilustra a operação de um CLP.

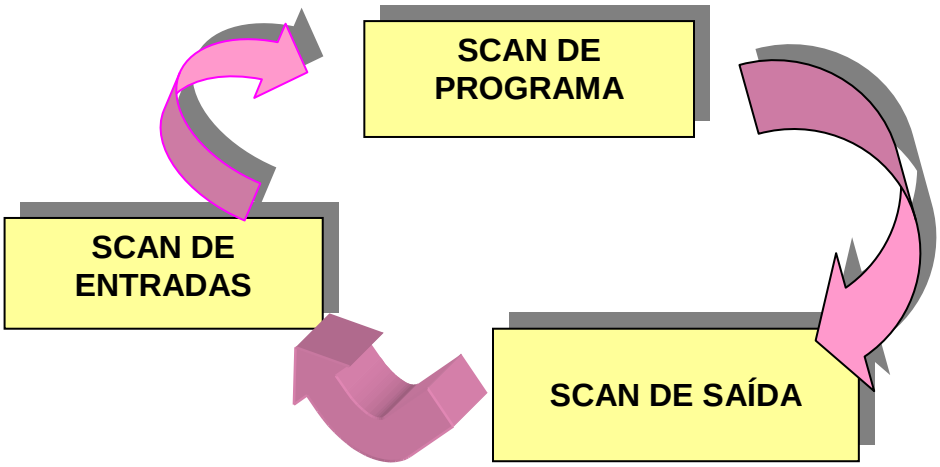


Figura 19 - Ciclo de operação de um CLP.

O tempo de SCAN total depende da velocidade do CLP empregado na automação do sistema e do tamanho do programa de aplicação do usuário. Determinadas instruções dos CLPs por serem mais complexas necessitam de um tempo maior de processamento. Os valores típicos de SCANs dos CLPs comerciais variam de 1 ms até 10 ms para um conjunto de 1000 instruções típicas. Assim sendo, nos projetos de automação de sistemas, o engenheiro deve conhecer antecipadamente qual o período de ciclo do processo que se deseja controlar e o tempo do SCAN do CLP. É necessário que o tempo total do SCAN do CLP seja significativamente menor que o tempo do ciclo do processo para que seja possível atingir as condições básicas de controlabilidade.

Entretanto, para determinar se o desempenho do sistema de automação atende as necessidades de um determinado processo ou não, também devem ser considerados os tempos de atualizações dos módulos de entrada e saída. Esses dados são importantes tanto para os processos de intertravamento, como para os processos de controle em malha fechada, quando o CLP utiliza instruções ou módulos de controle PID (Proporcional, Integral, Derivativo).

No SCAN de programa as instruções do programa são executadas seqüencialmente (ou seja, uma instrução depois da outra na seqüência direta de como foram carregadas no programa), portanto a ordem de como elas são introduzidas no programa pode alterar o número de SCAN necessários para se obter uma determinada condição ou até mesmo a própria condição final de uma determinada lógica.

A figura 20 ilustra um exemplo de como o resultado lógico final do intertravamento também pode eventualmente divergir conforme a sequência de programação foi alterada.

A figura 21 ilustra um exemplo de como no número de SCANs necessários para que uma variável de saída seja alterada em função de uma alteração de uma variável de entrada pode ser maior em função de como o programa do usuário foi editado. Tanto na figura 20, como na figura 21 os exemplos os programas estão escritos na linguagem de programação Ladder.

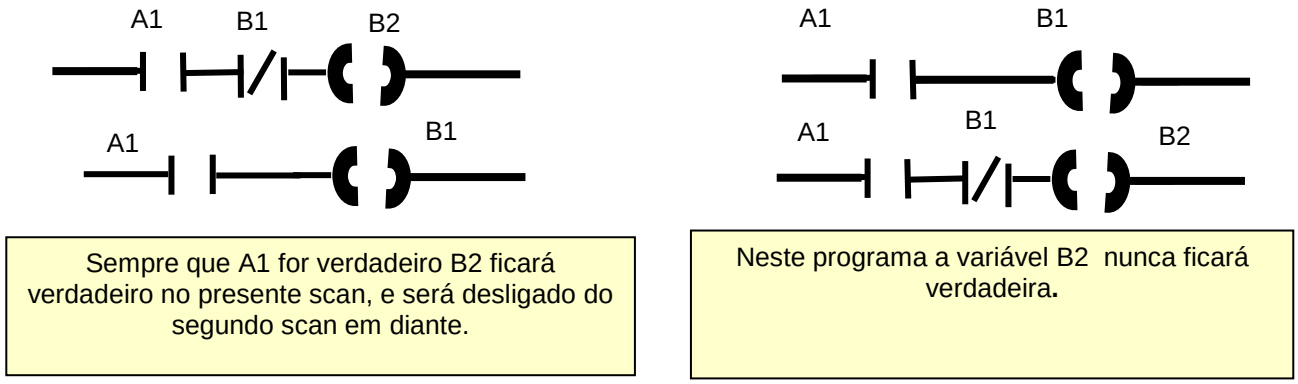


Figura 20 - Ilustrativo de alteração de performance em função da seqüência de instruções introduzida.

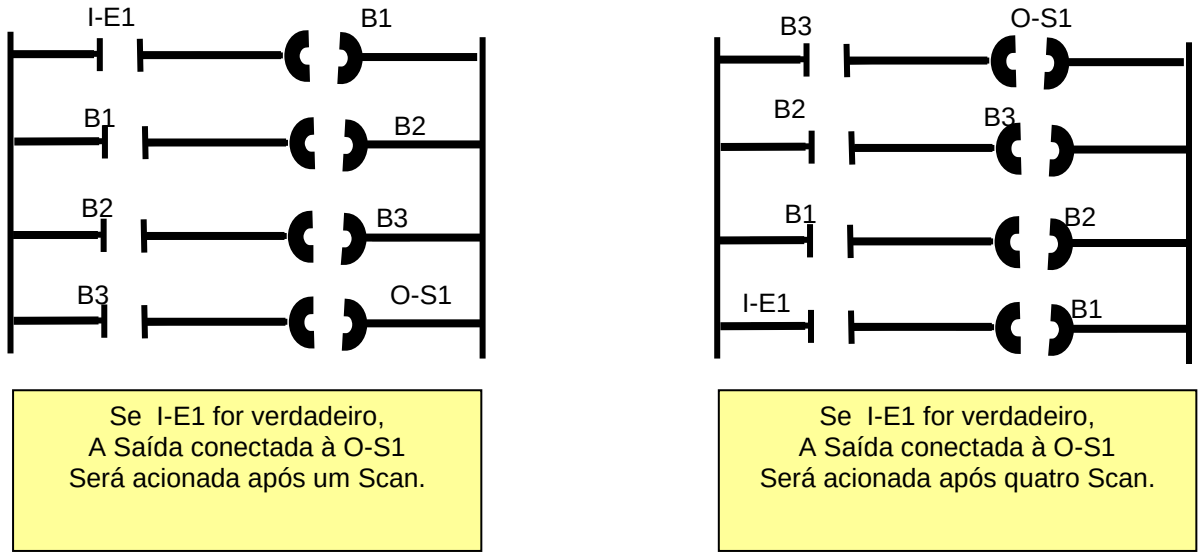


Figura 21 - Ilustrativo de alteração de desempenho em função da seqüência de instruções introduzida.

2.4.2 Endereçamento de memória dos CLPs

A capacidade de memória e a filosofia de endereçamento dos CLPs variam de modelo para modelo e de fabricante para fabricante. Porém, qualquer CLP deve ter uma área de sua tabela de dados que represente uma imagem virtual das entradas ligadas aos cartões de entrada, e uma área da tabela de dados que represente uma imagem virtual das saídas ligadas aos cartões de saída.

Como geralmente os CLPs são modulares, ou seja, a sua configuração pode ser expandida dentro de certos limites, estas áreas podem também variar respeitando a filosofia de projeto do fabricante. Essas áreas são normalmente designadas como imagem das entradas e imagem das saídas. Alguns fabricantes especificam quais são os endereços válidos para cada tipo de função, outros permitem uma maior flexibilidade. Porém qualquer que seja o modelo e sistema de numeração empregado no endereçamento, a filosofia dos diversos CLPs é parecida.

Muitos CPs utilizam palavras de 16 bits chamadas de registradores ou registros, e variáveis binárias de um bit, chamadas de pontos. Entretanto, já existem no mercado CLPs que utilizam palavras de 32 bits. Assim sendo, a codificação por exemplo em Diagrama Ladder, de um endereço de um CLP genérico poderia ser.

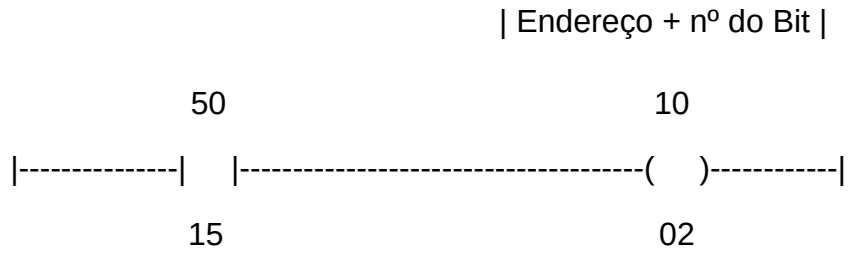


Figura 22 - Exemplo de endereçamento.

O exemplo ilustrado na figura 22 significa que o registro número (endereço) 50, bit 15, se for levado para condição 1 (verdadeiro) acionará a bobina representada pelo registro número (endereço) 10, bit 02. Portanto, ao olhar para o número do Ponto, o usuário pode determinar o registrador que contém tal dado e o respectivo bit dentro daquele registrador. Caso seja desejado registrar valores de temporizações, contagens, valores estatísticos, etc., ao invés de campos de bits (bit maps) tais registradores poderão

ser considerados, por exemplo nos sistemas de 16 bits, como variáveis inteiras de 16 bits, podendo armazenar valores entre -32768 a +32768.

Exemplo: suponha que um circuito elétrico tradicional esteja comandando a partida de um motor trifásico. A figura 23 representa um circuito elétrico de comando de acionamento desse motor.

Esse circuito emprega:

a) uma botoeira normalmente fechada;

b) uma botoeira normalmente aberta;

c) um contator para acionar o motor trifásico;

d) um contato auxiliar (normalmente aberto) do contator.

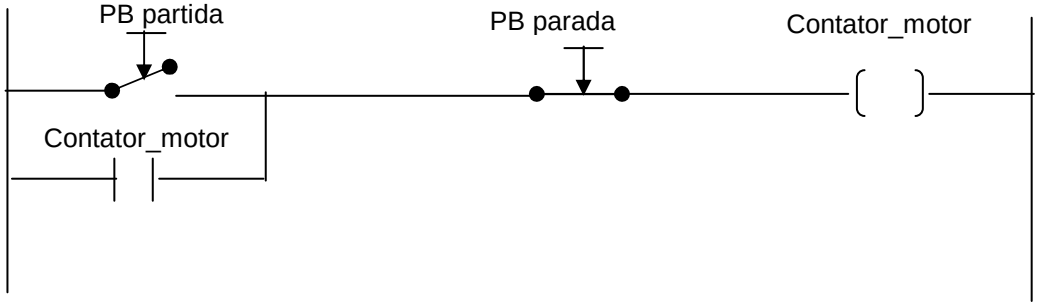


Figura 23 - Circuito de comando genérico para acionamento de um motor trifásico.

Para substituir esse circuito de acionamento por um sistema de acionamento que empregue um CLP, será necessário empregar duas entradas (uma para cada botoeira), e uma saída para ser ligada à bobina do contator. No caso de estar se empregando um CLP modelo SLC5000 da Rockwell Automation, por exemplo, o circuito seria:

a) Botoeira de partida (PB) na entrada 0 do cartão de entrada digital;

Obs. Se o cartão for colocado no Slot 1 do rack o endereço da mesma será I:1/0.

b) Botoeira de parada (PP) na entrada 1 do cartão de entrada digital;

Obs. Se o cartão for colocado no Slot 1 do rack o endereço da mesma será I:1/0.

c) Na saída 0 do cartão de saída a relé a ligação para a bobina do contator;

Obs. O endereço será O:2/0 se o cartão de saída estiver no Slot 2 do rack.

Neste caso, o diagrama ladder poderá ser como ilustrado na figura 24.

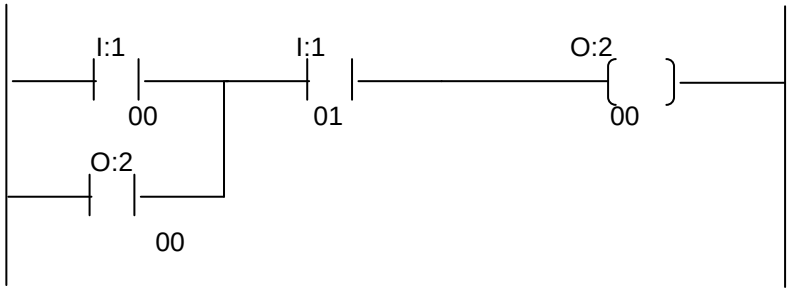


Figura 24 - Diagrama ladder para o acionamento do motor.

2.5 Controlador de Automação Programável (CAP) ControlLogix

O CAP ControlLogix (CLX), assim como os demais CAPs, são controladores que apresentam a elevada confiabilidade de hardware dos CLPs e a elevada capacidade de processamento e quantidade de memória dos microcomputadores. O objetivo principal no projeto de um CAP é que o mesmo possua as funções de um CLP e de um SDCD (Sistema Discreto de Controle Distribuído) com o preço e a simplicidade do CLP. Assim um CAP pode executar programas complexos, elaborados em linguagens de alto nível, conforme as necessidades dos diversos sistemas de automação. Os principais fabricantes e fornecedores de sistemas de automação industrial baseados em PAC's são:

a) GE – séries RX7i e RX3i;

b) Rockwell Automation – família ControlLogix;

c) Siemens – família SIMATIC S7;

d) Schneider – família Modicon M340;

e) ABB – sistema Compact Products 800.

Enquanto que os principais fabricantes e fornecedores de sistemas de automação industrial baseados em SDCD's são:

a) ABB – sistemas 800xA e Freelance 800F;

b) Emerson – sistema DeltaV;

c) Honeywell – sistemas Experion e PlantScape;

d) Yokogawa – sistemas CENTUM VP, CENTUM CS, CS 3000R3 e CS 1000R3;

e) Siemens.

Convém, apenas a título de recordação, ressaltar que os sistemas automatizados de monitoração e controle com controladores programáveis (CP) podem ser baseados em:

a) SDCD (Sistema Digital de Controle Distribuído);

b) CLP + SCADA (Controlador Lógico Programável + Supervisory Control And Data Acquisition);

c) Sistema Híbrido.

A figura 25 apresenta uma fotografia do CAP ControlLogix Observa-se que a fotografia apresenta um rack contendo uma fonte (à esquerda), diversos cartões e dois CAP instalados no mesmo rack. Na figura, cada CAP pode ser identificado pela chave presente em seu painel frontal, que permite escolher o modo de operação de cada controlador (comentado adiante no documento). Este modelo também permite que os dois CAPs distintos acessem todos os cartões de entrada instalados em um mesmo rack comum.

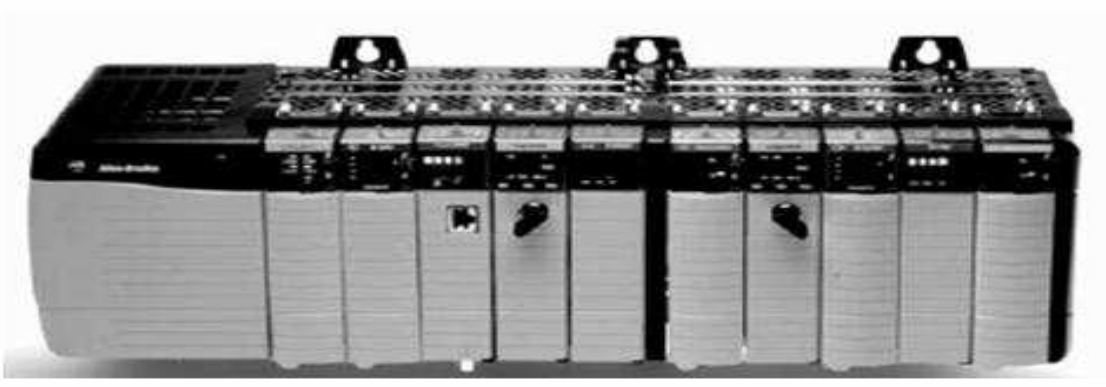


Figura 25 - Fotografia do CAP ControlLogix.

A arquitetura deste modelo difere das arquiteturas dos outro modelos de CLPs da Rockwell Automation. Nesta arquitetura existem pelo menos duas UCPs: a denominada UCP Logix e a UCP de “backplane”. *Backplane* é a placa onde tanto os CAPs, CLPs e cartões de Entrada/Saída e de comunicação são conectados. para o controlador mostrado, a “backplane” é a placa traseira do rack.

A UCP Logix executa os aplicativos e programas, além de enviar as mensagens nos barramentos de comunicação, conforme a necessidade.

A UCP de “backplane” se comunica com os cartões de entrada e saída, e opera de forma independente da UCP do Logix.

Esta arquitetura permite uma maior versatilidade de operação, uma vez que as duas UCPs operam de forma assíncrona e independente. A figura 26 ilustra a macro arquitetura das memórias de lógica/dados. De entradas/saídas e as duas UCPs.

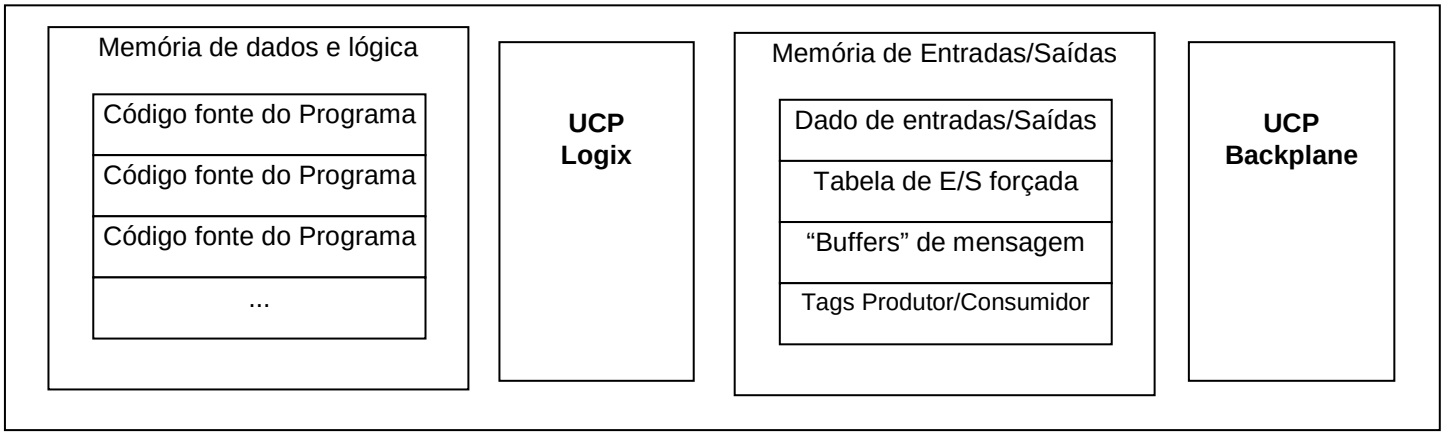


Figura 26 - Ilustração das duas memórias e UCPs da arquitetura ControlLogix

A arquitetura do ControlLogix emprega a estrutura produtor/consumidor, o que permite que as informações de entradas e os estados das saídas possam ser divididas e acessadas por diferentes Controladores Logix, localizados no mesmo ou em outros racks. A figura 27 ilustra esta possibilidade, empregando a mesma fotografia mostrada na figura 25.

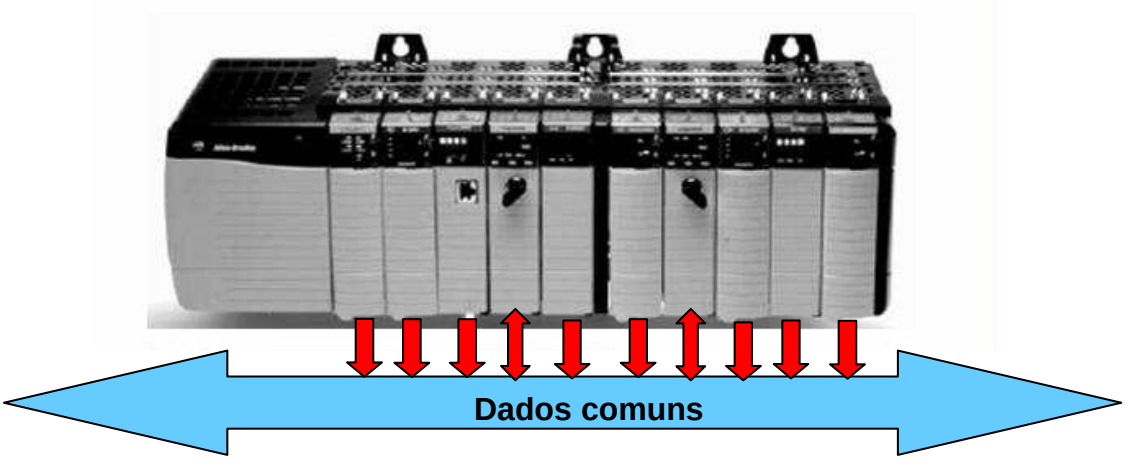


Figura 27 - Ilustrativo da atuação produtor/consumidor da arquitetura do ControlLogix

Observa-se que na arquitetura do ControlLogix existem dois tipos de Scan assíncronos: O SCAN de programa (de cada controlador) e o SCAN de entradas e saídas. A figura 28 ilustra os dois tipos de Scan quando em um rack existem um ou mais cartões de entrada/saída e um controlador Logix. A tabela 1 transcreve algumas informações importantes existentes no material do curso de treinamento em manutenção em ControlLogix.

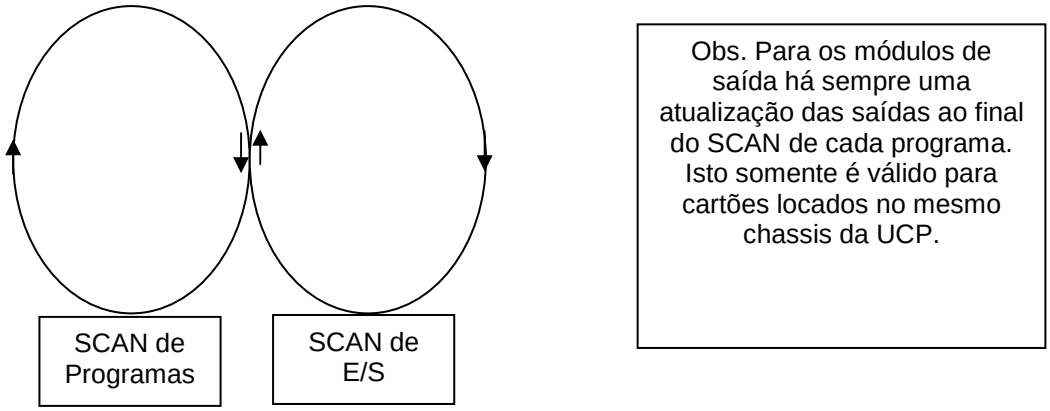


Figura 28 - Ilustrativo dos dois tipos de SCAN na arquitetura ControlLogix.

A tabela 1 a seguir mostra informações e conceitos importantes, como citado no no material dos cursos de treinamento em manutenção em ControlLogix.

Tabela 1 – Conceitos associados aos CAPs ControlLogix.

Atualização das Entradas e Saídas	A troca de dados entre os dispositivos de E/S e o Controlador obedece ao modelo produtor/consumidor. Desta forma a varredura de entrada e atualização das saídas não está necessariamente atrelada ao SCAN.
RPI – Request Packet Interval	Especifica a taxa na qual os dados são produzidos por um cartão de entrada ou saída. Este valor está compreendido entre 0,2 a 750 milisegundos.
COS – Change of Sate Somente para módulos digitais	Um módulo/cartão de entrada produzirá informação somente quando houver uma transição de ON para OFF ou OFF para ON, detectada pelo circuito de entrada.
RTS – Real Time Sample Somente para módulos analógicos de entrada	É o tempo gasto para executar as seguintes ações: ler todos os canais de entrada, atualizar status e zerar o contador do RTS.
Endereçamento	No ControlLogix não existe uma tabela de E/S pré-definida. Esta tabela é formada conforme os módulos são configurados. Endereçamento Local: é quando o módulo está no mesmo rack que a UCP do ControlLogix. Endereçamento Remoto: è quando o módulo não está no mesmo rack que a UCP do ControLogix.
Conexões É um enlace entre dois dispositivos. Estes dispositivos podem ser controladores, módulos de comunicação, módulos de E/S, variáveis produzidas e consumidas ou mensagens. O ContrlLogix suporta até 250 conexões.	Conexão Direta: quando cada cartão consome uma conexão.
	Conexão Rack Otimizado: quando cada rack consome apenas uma conexão.
	Conexão Mista: (direta + rack otimizado): quando existem módulos analógicos em um rack remoto, porque cada cartão analógico obrigatoriamente consome uma conexão.
Compartilhamento de E/S Os cartões de E/S do ControlLogix podem ser compartilhados de três formas.	Multicast: quando mais de uma UCP pode ser proprietária de um único cartão: Este compartilhamento funciona apenas para cartões de entrada que estejam configurados da mesma forma em todas as UCPs.
	Owner: quando somente uma única UCP pode ser proprietária de um cartão. Os cartões de saída só podem ser configurado em uma UCP como proprietário, pois não é possível que duas UCPs escrevam no mesmo cartão de saída.
	Listen Only: quando uma UCP só pode ler os dados de um cartão, não podendo escrever ou configurar o mesmo. Um cartão de saída deve ser configurado em uma UCP como *owner*. Nas outras UCPs tal cartão deve ser configurado como *Listen Only*.
Funções básicas do controlador	Executar comandos programados, ler entradas, acionar as saídas, trocar dados com outros controladores e comunicar-se com dispositivos remotos.

Normalmente existem três posições (estados) que definem o modo de operação de um CLP ou CAP. As mesmas são:

I. Posição RUN: Esta posição habilita o CLP no modo de operação (Run). O CLP varre/executa as tarefas do programa, monitora dispositivos de entrada, energiza dispositivos de saída e ativa pontos forçados de E/S habilitados. O modo do CLP pode ser alterado somente por meio de uma chave seletora posicionada no painel frontal do módulo da UCP Logix. Não é possível desenvolver a edição do programa online.

II. Posição PROG: Esta posição habilita o CLP no modo de programação (Program). O controlador não varre/executa as tarefas do programa e as saídas são desligadas. É possível desenvolver a edição do programa online. O modo do CLP pode ser alterado somente por meio de uma chave seletora posicionada no painel frontal do módulo da UCP Logix.

III. Posição REM: Esta posição habilita o CLP no modo remoto (Remote), quando podem ser designados remotamente os modos de operação REMote Run, REMote Program ou REMote Test. Tais modos de operação podem ser alterados por meio da posição da chave seletora ou através da interface de programação/operação. É possível desenvolver a edição de programa online nessa condição.

2.5.1 Organização de Memória no CAP ControlLogix

Enquanto que a maioria dos CLPs executa programas que podem ou não conter outros subprogramas de controle e aplicações diversas, na estrutura do ControlLogix (CLX) a aplicação geral desenvolvida é denominada de **Projeto,** que contém elementos de aplicação divididos em: tarefas (tasks), programas e sub-rotinas. O CLX possui internamente um sistema operacional, em tempo real, com recursos de multitarefas, com prioridades. Nesse contexto, os três conceitos (tarefas, programas e subrotinas) são fundamentais para o entendimento de seu funcionamento.

A tabela 2 apresenta uma síntese dos conceitos de tarefas (tasks), programas e sub-rotinas. A tabela 3 e a figura 29 apresentam um exemplo de tabela de prioridade, e de como as tarefas são executadas segundo as prioridades pré-definidas.

Tabela 2 - Resumo dos conceitos sobre: Projeto, Tarefas, Programas e Rotinas.

Projeto	É a aplicação completa. É o arquivo que armazena a lógica, as configurações, os dados e a documentação para o controlador e para o projeto de automação.
Tarefas (Tasks)	Uma tarefa (task) é o mecanismo de escala de execução de um programa. Uma aplicação pode ser dividida em muitas *tasks*. As *tasks* possibilitam que haja uma escala de operação, e também de prioridade, para as tarefas que devem ser executadas pelo controlador. Existem três tipos de *tasks*: a – Tarefas (*Tasks*) contínuas: Estas *tasks* são executadas continuamente, a menos que uma *tasks* periódica ou baseada em evento seja acionada. b – Tarefas (*Tasks*) periódicas: São *tasks* que são executadas em intervalos de tempo definidos. A taxa de uma *task* periódica pode ser de 0,1ms à 2.000 segundos. c – Tarefas (*Tasks*) baseadas em eventos: São executadas apenas quando um evento especificado ocorre.
Programa	Um programa pode ser definido como um conjunto relacionado de rotinas e tags. Um programa possui uma ou mais rotinas ou sub-rotinas.
Rotinas	É um conjunto de instruções lógicas escritas em uma das linguagens de programação.

Tabela 3 - Exemplo de tabela de especificação de tarefas e prioridades

Tarefa	Tipo de Tarefa	Nível de prioridade	Tempo de Execução
1	Periódica de 10 ms	5	2 ms
2	Periódica de 20 ms	10	4 ms
3	Contínua	nenhum	24 ms
4	Evento	Máxima ou configurável	Quando o evento ocorrer

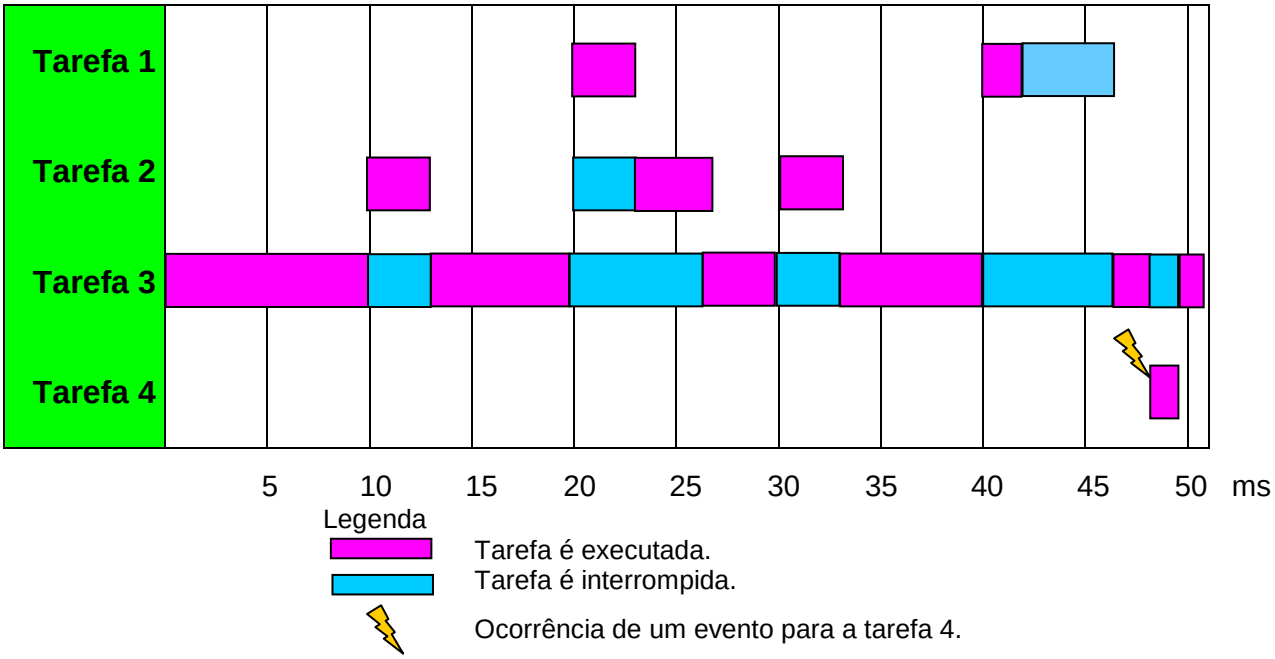


Figura 29 - Exemplo de execução de tarefas em função do tipo e da prioridade.

Dá Fig. 29 é importante observar que:

a) Todas as tarefas periódicas interrompem a tarefa contínua.

b) A tarefa de maior prioridade interrompe todas as tarefas de menor prioridade.

c) Quando uma tarefa contínua termina a mesma é reiniciada.

d) Tarefas com a mesma prioridade são executadas por meio da divisão de tempo entre elas com intervalos de 1 ms.

e) Tarefas associadas a eventos podem interromper as tarefas de maior prioridade ou disputar o processamento com outras tarefas escalonadas.

A figura 30 ilustra um exemplo da estrutura de organização de um projeto implementado em um ControlLogix para uma máquina que produz o empacotamento de um material. Observa-se que este tipo de máquina requer diversos sistemas para movimentação, controle de velocidade, controle de fluxo, controle de temperatura, bem como diversos sistemas de controle de qualidade do empacotamento do material. Assim, diversas tarefas, de diferentes tipos e com diferentes níveis de prioridade, poderão ser implementadas para uma melhor eficiência do controle automático.

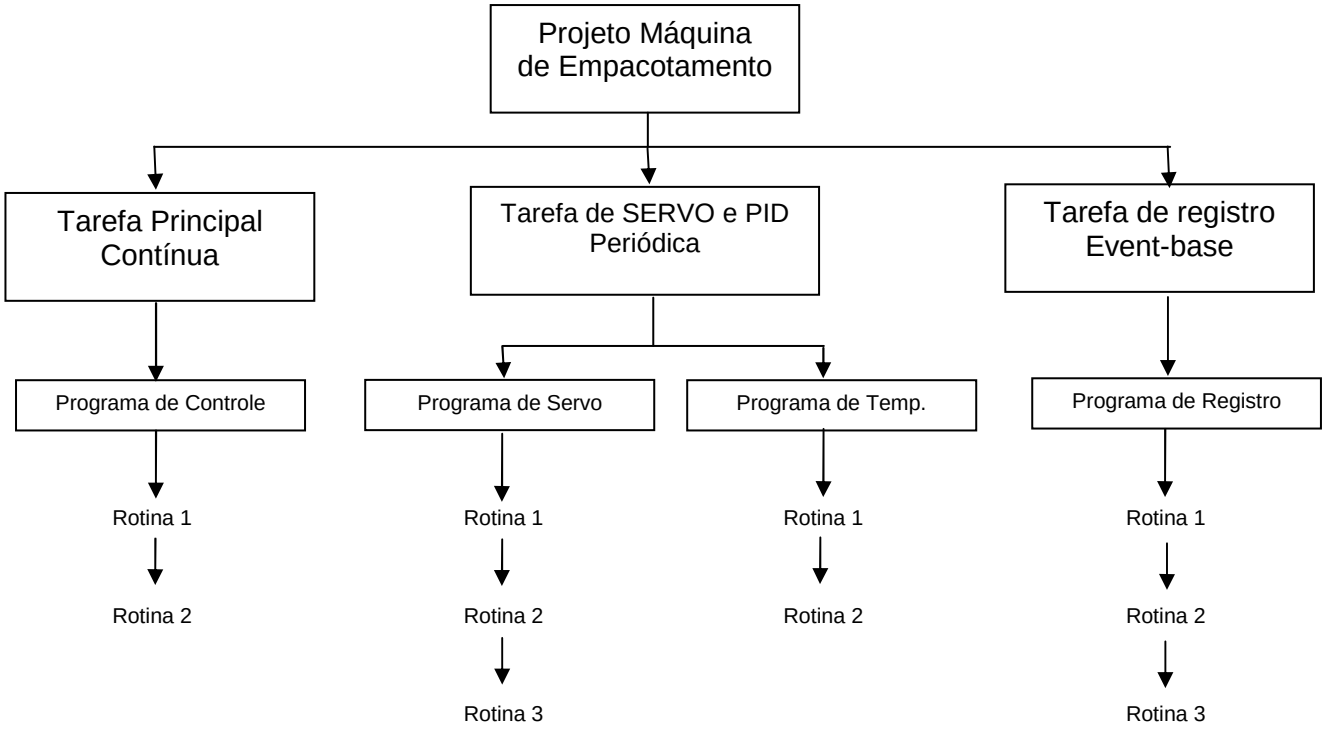


Figura 30 - Ilustrativo da estrutura de organização de um projeto implementado em um ControlLogix para uma máquina que produz o empacotamento de material.

2.5.2 Dados e endereçamento no CLX: TAGs, Alias, Array e ADD On Instruction

A seguir são apresentados conceitos sobre o endereçamento e tipos de dados disponíveis para um projeto em um CLX, além de outros conceitos relacionados.

Uma TAG é um nome amigável para o endereçamento de um local específico da memória. Por exemplo, a palavra “Temp” poderia ser um bom nome para a TAG que irá armazenar na memória o valor da temperatura lida por um sensor.

No CLX, o endereçamento das entradas, saídas, bits, variáveis internas, temporizadores, contadores, etc. são efetuados por meio de TAGs. O processador emprega o nome da TAG para acessar o endereço físico do dado na memória. Dessa forma o CLX não necessita, como nos demais processadores, de uma referência cruzada com o endereço físico de uma entrada ou saída. O nome da TAG identifica o dado completamente e, além disso, permite que a documentação do programa seja efetuada de forma clara na representação da aplicação. No caso do CLX, observa-se que a memória mínima ocupada por uma TAG são 4 bytes (ou 32 bits) para o dado em si, e mais 40 bytes para o nome da própria TAG.

Em um projeto com um CLX existem quatro tipos de TAGs: Base, Alias, Produced e Consumed. A tabela 4 ilustra esses tipos de TAGs.

Tabela 4 - Tipos de TAGs no CLX.

Tipo de TAG	Emprego do tipo de TAG
Base	Armazena tipos de valores para uso lógico no projeto.
Alias	Representa outra TAG.
Produced	Envia dados para outro controlador.
Consumed	Recebe dados de outro controlador.

Uma Base TAG armazena qualquer tipo de dado para ser empregado na lógica do projeto, tais como: bit, inteiro, etc. A tabela 5 ilustra os tipos de Base TAGs e os respectivos empregos para cada tipo. A tabela 6 ilustra os bits empregados no armazenamento dos valores para cada tipo de Base TAG.

Tabela 5 - Tipos de Base TAGs e os respectivos empregos para cada tipo.

Tipo de TAG	Emprego
BOOL	Bit
BOOL	Pontos de entrada e saída (E/S) digitais
CONTROL	Sequenciadores
COUNTER	Contadores
DINT	Inteiros longos (com 32 bits)
INT	Dispositivos analógicos em modo inteiro (Taxa de varredura rápida)
SINT	Inteiro curto (8 bits)
REAL	Numeros com ponto flutuante
TIMER	Temporizadores

Tabela 6 – Exemplo dos bits empregados no armazenamento dos valores para cada tipo de Base TAG.

Tipo de TAG	Uso do Bit e tamanho do número para cada tipo			
	31 16	**15 8**	**7 1**	**0**
BOOL	Não usado	Não usado	Não usado	0 ou 1
SINT	Não usado	Não usado	- 126 até 127	
INT	Não usado	-32.768 até 32.767		
DINT	-2.147.483.648 até 2.147.483.647			
REAL	-3,40282347E38 até – 1,17549435E-38 (valores negativos) Ou 1,17549435E-38 até 3,40282347E38			

Uma TAG tipo Alias é usada para criar um nome alternativo, ou seja um “apelido” para uma TAG já existente e já nomeada. Um Alias é um símbolo para um endereço específico de entrada ou saída. Esse nome pode representar uma entrada ou saída do mundo real, ou seja, a entrada ou saída de um dispositivo de campo. Um Alias também pode ser interpretado como uma TAG indexada nela mesma, ou seja, ele está “linkado” ou relacionado a TAG base. Assim, qualquer ação na TAG base também acontece com o Alias e vice e versa. A figura 31 ilustra um exemplo em linguagem Ladder de nomes Alias associados aos seus endereços reais.

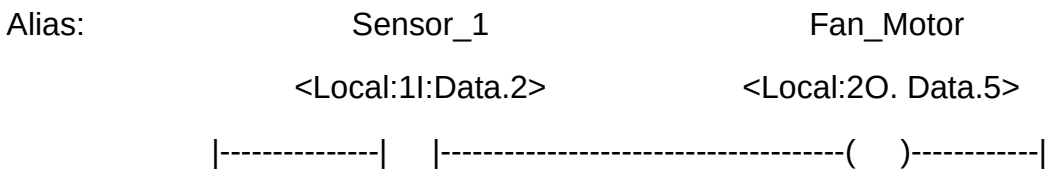


Figura 31 - Ilustração de uma linha de um programa em Ladder

O endereçamento das entradas e saídas E/S no CLX difere dos outros CLPs. As informações de E/S são apresentadas como um conjunto de TAGs. Cada TAG pode usar uma estrutura de dados particular. A estrutura depende das características específicas de cada módulo de E/S. O nome da cada TAG é baseado na localização do módulo de E/S no sistema. A figura 32 ilustra parte deste processo.

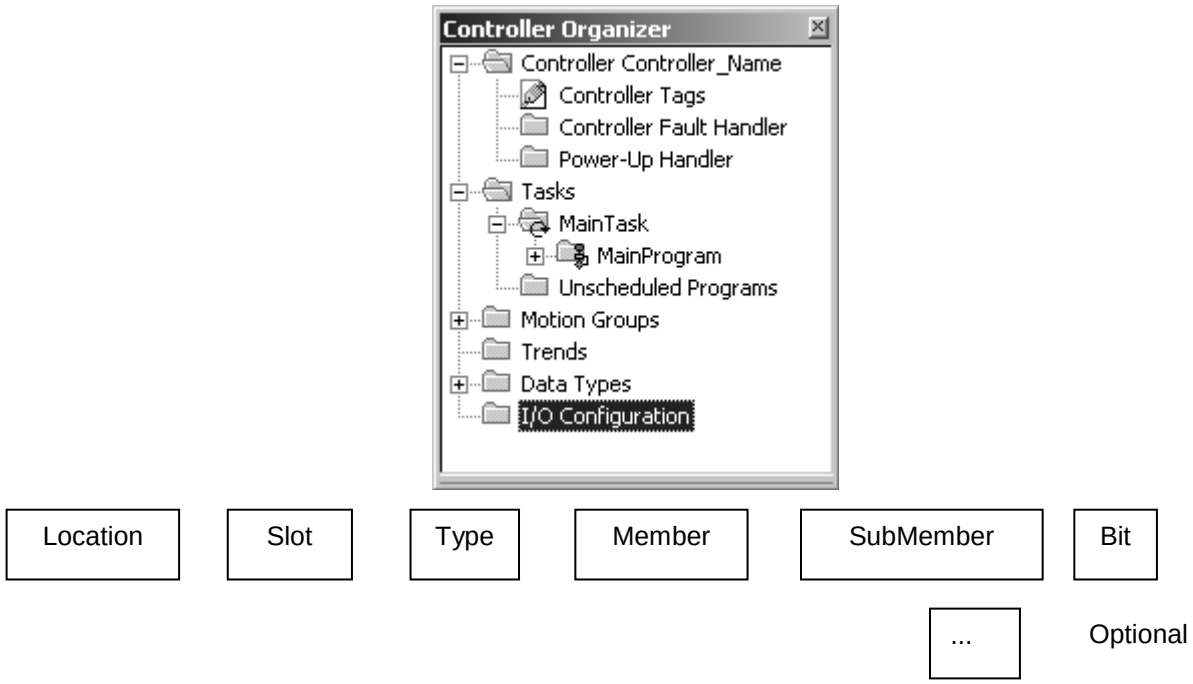


Figura 32 - Ilustrativo parcial do endereçamento das E/S, como visto na árvore de configuração do projeto no software de parametrização do ControlLogix.

A estrutura do endereçamento da Fig. 32 é descrita na tabela 7, que transcreve parte do manual Logix5000 Controllers I/O and Tag Data Catalog Numbers 1756 ControlLogix, 1756 GuardLogix, 1768 Compact GuardLogix, 1769 CompactLogix, 1789 SoftLogix, PowerFlex with DriveLogix Programming Manual Publication 1756-PM004C-EN-P - October 2009.

Tabela 7 - Transcrição parcial do manual Logix5000 Controllers I/O para endereçamento.

Where	**Is**
Location	Network location. LOCAL = same chassis or DIN rail as the controller ADAPTER_NAME = identifies remote comm. adapter or bridge module
Slot	Slot number of I/O module in its chassis or DIN rail
Type	Type of data: I = input O = output C = configuration S = status
Member	Specific data from the I/O module. Depends on what type of data the module can store. • For a digital module, a Data member usually stores the input or output bit values. • For an analog module, a Channel member (CH#) usually stores the data for a channel.
SubMember	Specific data related to a Member.
Bit	Specific point on a digital I/O module; depends on the size of the I/O module (0…31 for a 32-point module).

Um Array é um tipo de TAG que contém um bloco de muitos pedaços de dados. Um Array é similar a uma tabela de valores. Com um array de valores de dados, cada pedaço do dado individual é denominado elemento. Cada elemento de um array é do mesmo tipo de dado do restante dos elementos. Um array é uma matriz de elementos de memória que pode assumir até três dimensões. Estes elementos podem ser do tipo SINT, INT, DINT, REAL etc. Somente não pode ser do tipo BOOL.

ADD ON Instruction é uma nova ferramenta disponível na “versão 16” do software do ControlLogix, que cria instruções customizadas que podem ser utilizadas em um ou mais projetos.

Escopo das TAGS refere-se à acessibilidade de uma determinada TAG com relação a um ou mais programas. Quando é criada uma TAG, o usuário define se a mesma é uma TAG do escopo do controlador (“controller tag”) – disponível para todo o controlador e para quaisquer de seus programas (dado global), ou uma TAG do escopo de um programa (“program tag”) – disponível apenas para um programa específico (dado local).

As TAGs de programa possuem precedência sobre TAGs do controlador, caso possuam os mesmos nomes. De forma sumarizada, existem dois tipos de escopos no CLX:

a) TAG do controlador: um “controller scope TAG” esta disponível para todos os programa do projeto. Os dados das TAGs do controlador são também disponíveis para o mundo real, através de protocolos de comunicação com os sistemas SCADA, por exemplo.

b) TAG de Programa: as TAGs de escopo de programa “program scope TAGs” estão disponíveis apenas dentro dos programas em que foram criadas.

A figura 33 ilustra a organização de um projeto no CLX. A figura 34 ilustra um exemplo de dois programas (A) e (B) pertencentes de um mesmo projeto onde os dois programas tem acesso ao dados do “controller scope” e onde também nenhum dos dois programas podem acessar dados do outro programa.

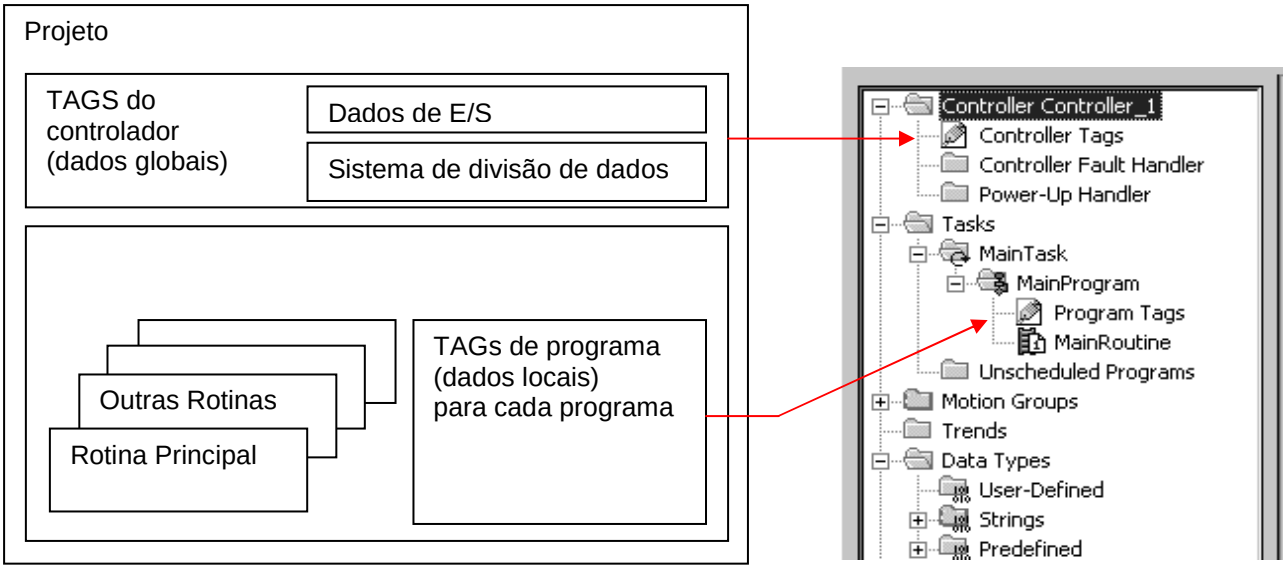


Figura 33 - Ilustração de um exemplo de organização de um projeto no CLX.

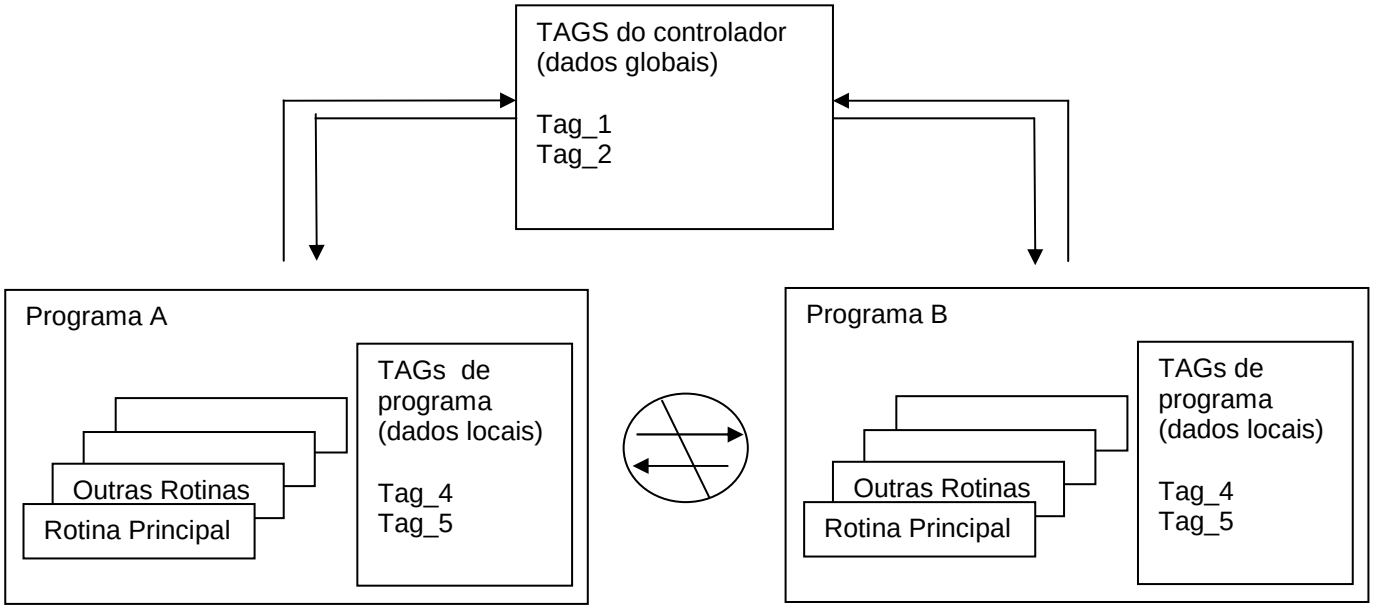


Figura 34 - Exemplo de dois programas com TAGs do Controlador e TAGs de Programa.

3 Linguagens de Programação de CLPs e CAPs

3.1 Norma IEC 61131-3

Por décadas a comunidade científica e diversas empresas do mercado de automação têm aplicado esforços no sentido de se padronizar equipamentos e linguagens de programação para os principais dispositivos e equipamentos empregados na automação de processos.

A *International Electrotechnical Commission* (IEC) é uma organização “irmã” da *International Organization for Standardization* (ISO), baseada em Genebra, na Suíça. Tais organizações possuem comissões e grupos de trabalho, com representantes da academia e das principais indústrias mundiais, para vários setores da engenharia. Dessas comissões, o comitê técnico TC65 da IEC foi o responsável pela normatização e padronização do controle e das medidas para processos industriais. Nesse comitê, o grupo 7 foi aquele estabelecido e encarregado por desenvolver uma norma para a padronização de CLPs e CAPs.

Como resultado das atividades desse grupo, em 1993, foi publicada a primeira versão da norma IEC 61131 sobre controladores programáveis (CPs). A norma consiste em 8 partes, inter-relacionadas, abordando aspectos específicos do tema.

Como mostrado na Tabela 8, os 5 principais tópicos da IEC 61131 abordam: informações e conceitos gerais; requisitos e testes para os equipamentos; linguagens de programação de CPs, diretrizes de usuário, e mensagens e comunicações. As demais partes abordam: as diretrizes para implementação das linguagens de programação, detalhes sobre programação com protocolos de comunicação de chão de fabrica (como o FieldBus) e detalhes para implementação de controle com lógica Fuzzy.

Tabela 8 - Escopo das partes principais da norma IEC 61131-3.

Parte	Título	Descrição
1	Informações gerais	Definições e conceitos básicos
2	Requisitos e testes para equipamentos	Construção elétrica e mecânica testes de verificação
3	Linguagens de programação	Estrutura de software, linguagens e execução de programas
4	Guia do usuário	Guia de seleção, manutenção e instalação. Padrões de projeto.
5	Especificações de serviços de mensagens e comunicações	Facilidades de software etc.

Além de promover em linhas gerais uma padronização para procedimentos, especificação, *hardware* e *software* para CPs, um dos principais objetivos da norma IEC 61131 é a melhoria e a garantia da qualidade do *software* desses controladores. Por exemplo, segundo a parte 3 da 61131 (conhecida como IEC 61131-3) um software em um CP possui os seguintes atributos principais:

Capacidade: Descreve a habilidade do sistema em executar as funções designadas e necessárias. A capacidade inclui: **responsividade**, **capacidade de processamento**, **capacidade de memória**.

Disponibilidade: Descreve a porcentagem de tempo em que um sistema está disponível. A disponibilidade inclui: "***reliability***" - que é o tempo que o sistema está efetivamente em operação, ou seja, o inverso do MTBF (*Mean Time Between Failures*); "***maintainability***" que é o inverso do tempo de reparo MTTR (*Mean Time To Repair*) e **integridade**.

Usabilidade: Descreve a facilidade do uso, ou o quanto o sistema é amigável. A usabilidade compreende: "***entry requirements***" que é relativo ao quanto de treinamento prévio os usuários devem ter para poder utilizar o sistema, "***learning requirements***" relativo a quanto de treinamento deverão ter os operadores do sistema para poder usá-lo, "***user productivity***" relativo ao número de operações possíveis, por usuário, em um determinado período de tempo, dentro de um determinado nível de treinamento, "***congeniality***" relativo à preferência do usuário em empregar um novo sistema com relação a sistemas anteriores.

Adaptabilidade: Descreve o quanto o sistema pode ser alterado ou adaptado a novas situações ou atualizações de *hardware* e de *software*. A adaptabilidade compreende: "***improvability***" que é relativo à capacidade de melhoria do sistema, "***extensibility***" relativo à capacidade de serem incorporadas novas funções, "***portability***" que é relativo à capacidade de mover ou portar a funcionalidade para outro sistema, e "***reusability***" que é relativo à capacidade de um elemento do *software* se empregado em outro sistema.

Nesses termos, a IEC 61131-3 foi desenhada para ser empregada, pelos diversos fabricantes, na elaboração de seus produtos de automação e na elaboração das linguagens de programação dos CPs, especificando os padrões e requisitos funcionais dessas linguagens e sistemas. Entretanto, como qualquer norma, os fabricantes podem aplicar as definições de forma fundamental, ou com um ou outro ajuste particular.

Segundo a IEC 61131-3, as principais linguagens de programação padronizadas são: Texto Estruturado (ST – *Structured Text*), Diagrama de Blocos Funcionais (FBD – *Functional Block Diagram*), Lista de Instruções (IL – *Instruction List*), Sequenciamento Gráfico de Funções (SFC – Structured Flow Control) e o tradicional diagrama LADDER.

No caso específico do CLX da Rockwell, um projeto pode ser conter um ou mais programas (com suas tarefas, rotinas e etc.), cada um elaborado em uma das quatro linguagens da IEC 61131-3. O Projeto é elaborado no software de desenvolvimento e parametrização de CPs da Rockwell Automation denominado RSLogix5000. O projeto é depois transferido para a memória de um CP de campo através do mesmo software, para poder ser executado no controlador.

Tais linguagens são descritas com detalhes a seguir.

3.2 Linguagem LADDER ou LD

Diagrama de “Ladder” ou Diagrama em Progressão “Ladder Diagram “(LD) é uma linguagem gráfica baseada em diagramas ou circuitos lógicos de relé. A figura 35 ilustra um exemplo de um trecho de programa de CLP escrito em linguagem Ladder.

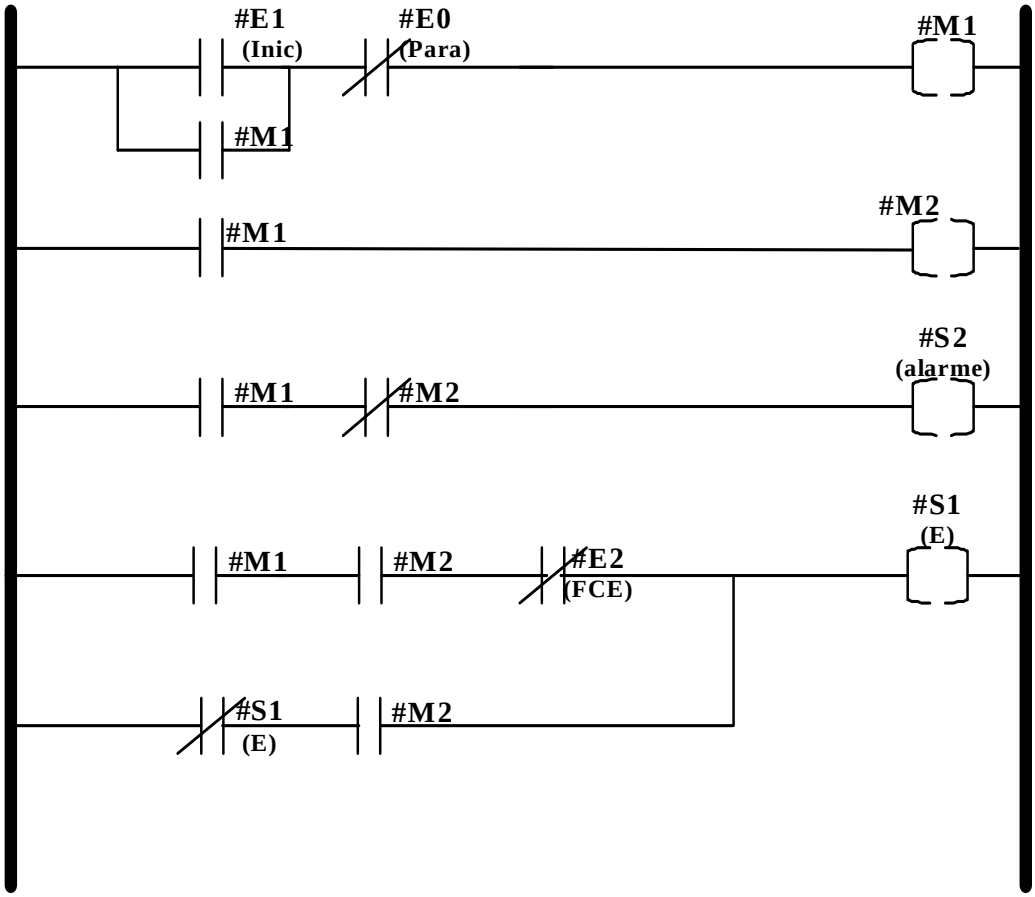


Figura 35 - Exemplo de um trecho de programa escrito em linguagem Ladder.

A linguagem LD é uma linguagem gráfica de alto nível que se assemelha ao esquema elétrico de um circuito de comando ou diagrama de contatos. No LD todos os tipos de instruções pertencem a dois grandes grupos: instruções de entrada e instruções de saída. Nas instruções de entrada são formuladas perguntas, enquanto que as instruções de saída executam algum tipo de ação em função das respostas afirmativas ou negativas das instruções de entrada, representadas na mesma linha lógica da instrução de saída.

A UPC do CP executa, caso não haja nenhum outro comando especificando algo em contrário, todas as instruções começando pela primeira instrução da primeira linha do programa, até a última instrução da última linha do programa do usuário.

Na linguagem LD os comandos imitam a estrutura de um esquema de circuito de intertravamento baseado em lógica de reles. Entretanto, é importante lembrar que apesar dessa semelhança com os tradicionais circuitos elétricos de comando e controle por relés, a lógica em LD não opera exatamente da mesma forma. Um programa de CP é composto basicamente de instruções de entrada e de instruções de saída. Durante uma instrução de entrada, a UCP verifica uma pergunta ou uma comparação. Caso a resposta seja afirmativa, é estipulado durante o SCAN atual uma continuidade lógica do trecho de linha analisado. Uma instrução de saída é processada conforme exista ou não continuidade lógica de linha até ela. Nesse esquema, uma linha do programa é executada por vez, na sequência apresentada, até que todo o programa seja percorrido, diferentemente das lógicas eletromecânicas a relés que possuem um caráter de execução intrinsecamente paralelo, ou seja, todas as linhas estariam sendo executadas simultaneamente.

As instruções básicas da maioria dos CPs podem ser agrupadas em sete grupos:

a) Lógica de rele ou instrução de Bit;

b) Temporização e contagem;

c) Aritméticas;

d) Manipulação de dados;

e) Controle de fluxo;

f) Transferência de dados, e;

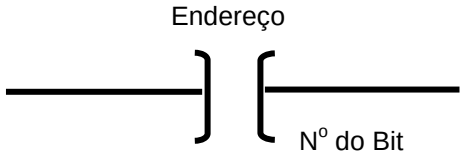
g) Avançadas.

Uma instrução de Bit pode ser de entrada ou de saída. Durante a execução de uma instrução de entrada, o estado de um Bit em um determinado endereço é examinado. Durante a execução de uma instrução de saída de bit, o estado de um bit de um determinado endereço é alterado para 0 ou 1 conforme haja ou não continuidade lógica da linha em que a instrução está relacionada.

Algumas das principais instruções de bit (de entrada e de saída), temporização e contagem são mostradas a seguir.

3.2.1 Intrução XIC → Examinar se energizado.

A UPC do CLP executa esta instrução verificando o valor do Bit endereçado pela mesma. Se o Bit endereçado estiver no estado lógico 0 a instrução retorna com o valor lógico falso e, portanto, não estabelece a continuidade lógica à direita no trecho do LD em que a instrução está inserida. Se o Bit endereçado estiver no estado lógico 1, a instrução retorna com o valor lógico verdadeiro e, portanto, estabelece a continuidade lógica no trecho do LD em que a instrução está inserida. A representação desta instrução na linguagem, juntamente com a sua operação, faz com que a mesma seja comumente interpretada como um contato normalmente aberto de um determinado relé. Porém, convém ressaltar que apesar do funcionamento ser análogo a mesma, assim como qualquer outra instrução de um CP, é uma instrução lógica e não um contato físico de um circuito elétrico. A figura 36 apresenta a representação da instrução XIC em linguagem LD e também a sua tabela verdade de operação



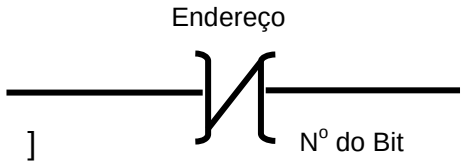
Estado do BIT	Instrução XIC
0	Falsa
1	Verdadeira

Figura 36 - Representação em linguagem Ladder da instrução XIC e a respectiva tabela verdade de operação.

3.2.2 Instrução (XIO) → Examinar se desligado.

A UPC do CP executa esta instrução verificando o valor do Bit endereçado pela mesma. Se o Bit endereçado estiver no estado lógico 1, a instrução retorna com o valor lógico falso e portanto não há continuidade lógica à direita no trecho do LD em que a instrução está inserida. Se o Bit endereçado estiver no estado lógico 0, a instrução retorna com o valor lógico verdadeiro e portanto é estabelecida a continuidade lógica à direita no trecho que a instrução está inserida. A representação desta instrução na linguagem LD, juntamente com a sua operação, faz com que a mesma seja comumente interpretada

como um contato normalmente fechado de um determinado relé. Entretanto, da mesma forma que a instrução anterior, convém ressaltar que apesar do funcionamento ser análogo a mesma, trata-se de uma instrução lógica e não de um contato físico de um circuito elétrico. A figura 37 apresenta a representação da instrução XIO em linguagem LD e também a sua tabela verdade de operação.



Estado do BIT	Instrução XIO
0	Verdadeira
1	Falsa

Figura 37 - Representação em linguagem LD da instrução XIO e sua respectiva tabela verdade de operação.

3.2.3 Instrução (OTE) → Energizar saída

A UPC do CP executa esta instrução verificando se há ou não continuidade lógica na linha que antecede essa instrução. Caso haja continuidade lógica, o Bit endereçado pela instrução será colocado no estado lógico 1. Se não houver continuidade, o Bit endereçado pela instrução será colocado no estado lógico 0. A figura 38 apresenta o aspecto gráfico da instrução OTE.

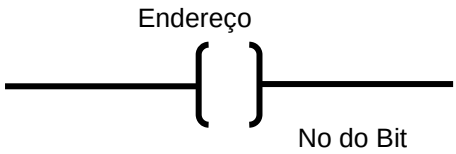


Figura 38 - Representação em linguagem LD da instrução OTE .

3.2.4 Instrução (OTL) → Energizar saída com retenção

A UPC do CP executa esta instrução verificando se há ou não continuidade lógica na linha que antecede a mesma. Caso haja continuidade lógica, o Bit endereçado pela instrução será colocado no estado lógico 1. Entretanto, uma vez habilitada a saída endereçada pela instrução, ou seja, uma vez que o Bit endereçado pela instrução OTL

seja colocado no valor lógico 1, o mesmo somente será desabilitado caso uma instrução OTU endereçada para o mesmo endereço do Bit da instrução OTL seja acionada. Em outras palavras a instrução OTL opera como um “selo” de um circuito de relé ou como um flip-flop do tipo Set-Reset (a instrução OTL faz o SET e a instrução OTU faz o RESET). A figura 39 ilustra a instrução OTL na linguagem LD.

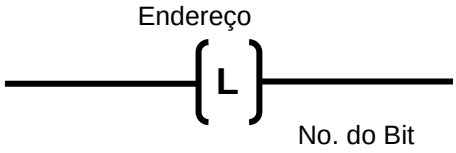


Figura 39 - Representação em linguagem LD da instrução OTL.

3.2.5 Instrução (OTU) → Desabilitar saída com retenção

A UPC do CP executa a instrução OTU verificando se há ou não continuidade lógica na linha que antecede a mesma. Caso haja continuidade lógica da linha, a instrução OTU desabilita a saída habilitada pela instrução OTL relativa ao mesmo endereço de Bit. Ou seja, caso haja continuidade lógica na linha que antecede a instrução OTU o Bit endereçado pela instrução OTL relativa a instrução OTU é colocado no estado lógico 0. A figura 40 ilustra a instrução OTL na linguagem LD.

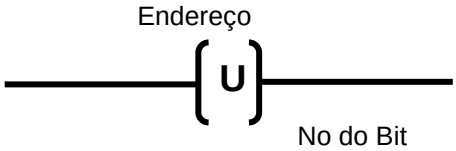


Figura 40 - Representação em linguagem LD da instrução OTU.

3.2.6 Instrução TON → Temporizador crescente sem retenção

A UPC do CP executa a instrução TON verificando se há ou não continuidade lógica na linha. Caso haja continuidade lógica, a instrução TON inicia uma contagem de tempo (“uma temporização”) baseada nos intervalos de tempo e na base de tempo selecionados durante a programação da instrução. A instrução TON possui Bits de controle e de sinalização do seu estado de operação.

O bit EN é colocado no estado lógico 1 cada vez que a instrução é acionada. O bit DN é colocado no estado lógico 1 quando o "valor do registrador acumulado" (valor ACCUM) for igual ao "valor do registrador pré-selecionado" (valor PRESET). A instrução "Temporizador crescente sem retenção" ocupa três palavras da memória do programa do usuário. A figura 41 ilustra a instrução TON na linguagem LD.

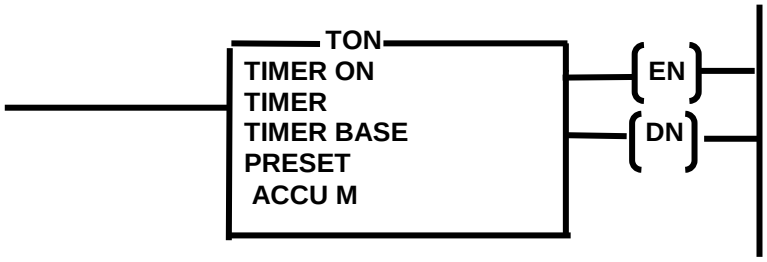


Figura 41 - Representação em linguagem LD da instrução Temporizador crescente sem retenção.

3.2.7 Instrução RTO → Temporizador Retentivo

A instrução de "Temporizador Retentivo", de maneira semelhante à instrução TON, é utilizada para "energizar" ou "desenergizar" um dispositivo, assim que for alcançado um valor de Preset. A instrução de Temporizador Retentivo, entretanto, retém o seu valor acumulado quando ocorrer qualquer uma das condições a seguir:

a) As condicionantes da linha passarem a falsas;

b) A chave seletora de modo for colocada na posição PROG;

c) Ocorrer falta de energia desde que seja mantida a energia de back-up da memória RAM.

Para zerar o temporizador, deve-se utilizar a instrução de rearme RTR.

3.2.8 Instrução CTU → Contador crescente

A UPC do CP executa a instrução CTU verificando se há ou não continuidade lógica na linha da mesma. A cada transição de falso para verdadeiro da condição lógica da linha em que a instrução está inserida, a instrução CTU incrementa o "valor do registrador acumulado" (valor ACCUM). Quando o valor ACCUM for igual ao "valor do registrador pré-selecionado" (valor PRESET), a instrução CTU coloca o Bit DN no valor lógico 1. A

instrução CTU ocupa três palavras da memória do programa do usuário. A figura 42 ilustra a instrução TON na linguagem LD.

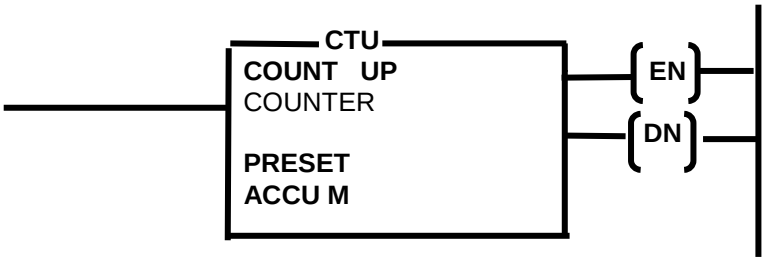


Figura 42 - Representação em linguagem LD da instrução contador CTU.

3.2.9 Instruções de Comparação e de Testar Limite

As instruções de comparação e de teste de limite são instruções de avaliam a relação entre dois valores: Origem A e Origem B:

- Origem A é o endereço de uma palavra.
- Origem B é o endereço de outra palavra ou de uma constante.

Com esses operandos, uma série de instruções de comparação e de teste de limites pode ser efetuada, como as mostradas a seguir. Sugere-se que o leitor procure na documentação técnica disponível as demais variações dessas instruções, tais como EQ, GEQ, LEQ, etc.

a) Instrução Menor Que → LES

A figura 43 apresenta o formato da instrução LES:

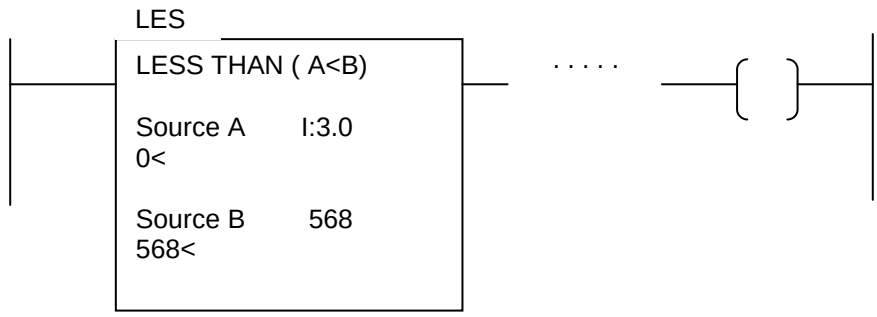


Figura 43 - Exemplo de Instrução Menor que

Quando o valor “Source A” for menor que o valor “Source B”, esta instrução será logicamente verdadeira. Quando o valor “Source A” for maior ou igual ao valor “Source B”, a instrução será falsa.

b) Instrução Maior que → GRT

A figura 44 apresenta o formato da instrução GRT.

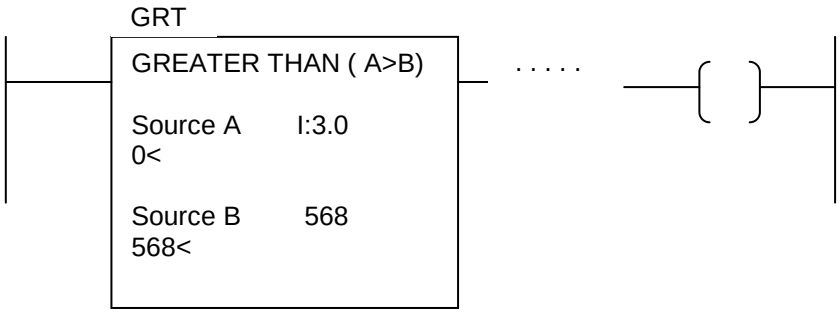


Figura 44 - Exemplo de Instrução Maior que

Quando o valor “Source A” for maior que o valor “Source B”, esta instrução será logicamente verdadeira. Quando o valor “Source A” for menor ou igual ao valor “Source B”, a instrução será falsa.

c) Instrução Testar Limite → LIM

Esta instrução de entrada testa os valores dentro ou fora de uma faixa especificada, dependendo de como foram ajustados os seus limites. A figura 45 mostra o formato da instrução:

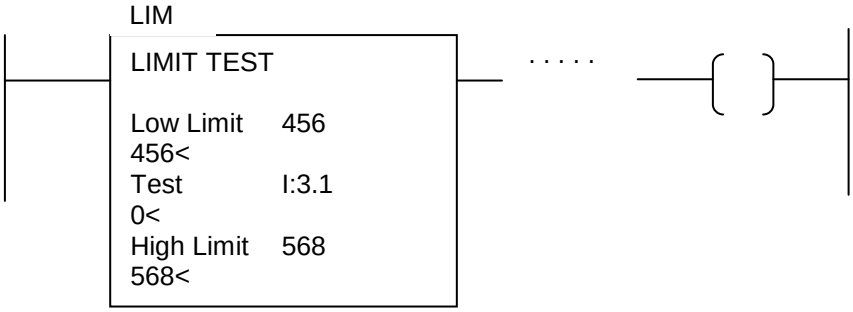


Figura 45 - Exemplo de Instrução Testar limite

Os parâmetros para esta instrução – Low Limit, Test e High Limit – podem ser constantes numéricas ou endereços de palavras que armazenam algum valor, conforme a tabela 9. A tabela 10 apresenta a funcionalidade da função da instrução LIM.

Tabela 9 - Exemplo de instrução testar limite.

Test	High Limit	Low Limit
Constante	Endereço de palavra	Endereço de palavra
Endereço de palavra	Constante ou Endereço de palavra	Constante ou Endereço de palavra

Tabela 10 – Funcionalidade da instrução LIM.

Se o limite inferior for:	A instrução LIM será verdadeira quando o valor de Teste estiver:
Menor que o limite superior	Entre os dois limites ou igual a um deles
Maior que o limite superior	Igual ou fora dos limites

3.2.10 Instrução OSR → Detector de borda de subida

O OSR é uma instrução que faz com que uma instrução de saída seja logicamente verdadeira apenas durante um ciclo de varredura do programa LD, quando sua entrada transita de falso para verdadeiro. Esta instrução deve ser posicionada sempre imediatamente anterior a uma instrução de saída, assim como representado na Figura 46.

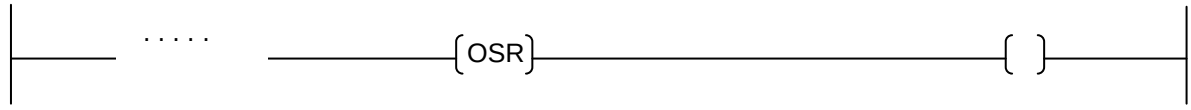


Figura 46 - Instrução OSR

É necessário que se atribua um endereço para a instrução OSR, que deve ser um bit reservado apenas para esta função. O endereço utilizado não deve ser utilizado em nenhum outro local do programa. Quando a condição da linha muda de falsa para verdadeira a instrução OSR se torna verdadeira, tornando verdadeira a condição de linha para a instrução de saída subsequente. Após um ciclo de varredura do programa LD, a instrução OSR se torna falsa, mesmo que as condições de linha à sua esquerda sejam verdadeiras. A instrução OSR apenas voltará a ser verdadeira quando houver, novamente, uma transição de falso para verdadeiro nas condições de linha à esquerda.

3.3 Linguagem SFC

O mapa ou diagrama sequencial de funções (*Sequential Function Chart*) ou SFC, é uma linguagem gráfica, baseada em Redes de Petri e GRAFCET, que descreve o comportamento sequencial de sistemas de controle, na forma de estados (ou passos) e transições entre esses estados. Tal linguagem é resultado da incorporação em 1988 da antiga norma IEC 848. A SFC pode ser empregada para descrever tanto sistemas no domínio do tempo como sistemas no domínio dos eventos discretos. A figura 47 ilustra um exemplo de um trecho de programa de CP escrito na linguagem SFC. Obs. Também existe o SFC descrito através de uma forma textual, não gráfica.

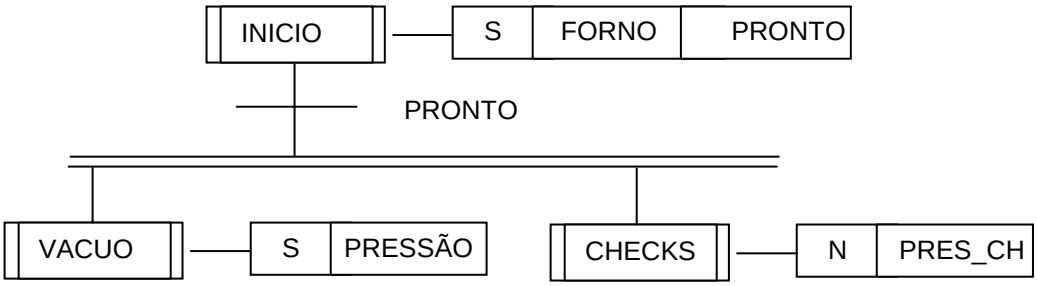


Figura 47 - Exemplo de um trecho de programa escrito em linguagem SFC.

A linguagem SFC emprega passos (ou estados) e transições para ilustrar a execução de suas operações ou ações. Normalmente a Linguagem SFC é conveniente quando o processo possui:

- Gerenciamento de alto nível de múltiplas operações;
- Sequências repetitivas de operações;
- Processos por Batelada;
- Controle tipo *Motion* (controle de movimento);
- Operações do tipo máquina de estados.

No SFC os retângulos representam os estados, passos ou estágios do sistema. As linhas de conexão entre os retângulos representam o fluxo ou a sequência de estados, enquanto que as barras nessas linhas de conexão representam as transições. Cada transição é associada a uma condição ou a um conjunto de condições booleanas. Dessa forma, no SFC um programa fica semelhante a um fluxograma ou máquina de estados.

O tempo de execução ou o fluxo de funcionamento desses diagramas de estado depende de condições estáticas (definidas pelo programa) e de condições dinâmicas (comportamentos das entradas/saídas). Cada passo ou unidade pode ser programado em qualquer das outras linguagens definidas pela norma IEC 61131-3 ou em outra descrição SFC.

Na SFC um passo está ou não ativo. Quando um passo está ativo, um conjunto associado de instruções (definido como ações) é executado repetidamente até o passo se tornar inativo. A decisão de alterar o status de um passo (ou seja, tornar um passo ativo ou inativo, ou vice-versa) é determinada pela transição, que é o elemento imediatamente seguido do passo. A transição é programada por meio de uma condição de transição, através de uma expressão booleana. Quando a expressão se torna verdadeira (VERDADEIRA/TRUE) o passo ativo é desativado e um próximo passo é assumido como ativo.

Na linguagem SFC, cada passo (estado) ou (lugar) é representado por um retângulo. O passo representa um determinado estado operacional do sistema e deve ter um único nome que o identifique. Existem dois tipos de passos: inicial e normal. A figura 48 ilustra os gráficos dos dois tipos de passo.

Figura 48 – Dois tipos de representação para os passos.

Quando um passo está ativo, o mesmo é assinalado por meio de uma marca “token”, ou por qualquer tipo de sinalizador que identifique o estado do passo. As variáveis associadas a um passo são:

- Variável FLAG, que indica que o passo está em atividade. Essa variável tem a nomenclatura (nome do passo).X.
- Variável TEMPO, com a nomenclatura (nome do passo).T, e que está associada à duração em tempo real, desde o início da atividade de um passo.

Em cada passo há um conjunto de ações que serão executadas quando o mesmo entra em atividade. A figura 49 ilustra a representação gráfica de um passo em SFC. A figura 50 ilustra um programa genérico simples para o controle de um reator em SFC. Observa-se que o bloco ações pode ser omitido.

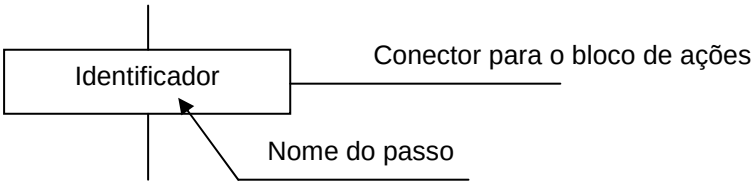


Figura 49 - Ilustração de um passo.

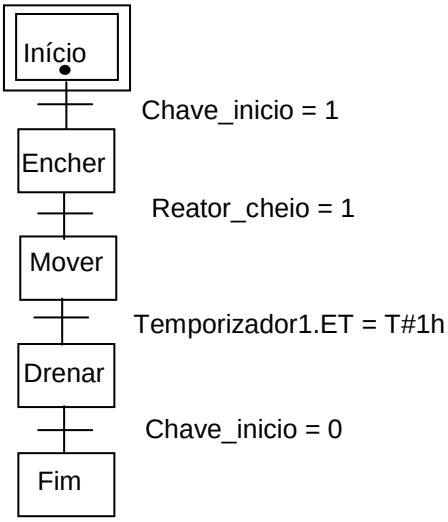


Figura 50 - Exemplo de um programa genérico simples para o controle de um reator em SFC.

3.3.1 Regras de evolução do SFC e do GRAFCET

A transição é uma barreira entre os passos. Ela impede o fluxo de execução até que sua condição seja satisfeita e até que a própria transição esteja habilitada. Uma transição está habilitada se todos os estágios ou passos anteriores estão ativos.

As condições de uma transição podem ser expressões lógicas, temporais, aritméticas, etc. e são denominadas de receptividade da transição. Na ocorrência de uma transição, ocorre a ativação de todos os estágios ou passos imediatamente posteriores e que estão conectados a esse transição. Numa transição também são desativados todos os estágios imediatamente precedentes e a ela conectados. A tabela 11 apresenta as principais regras de transição na linguagem SFC, conforme a IEC 61131-3. A tabela 12 apresenta a ilustração de três exemplos de sequência.

Tabela 11 - Principais regras de transição na linguagem SFC conforme a IEC 61131-3.

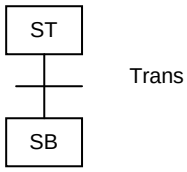
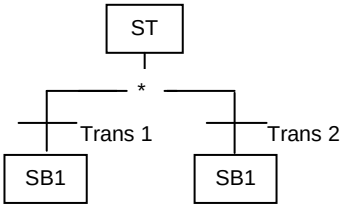
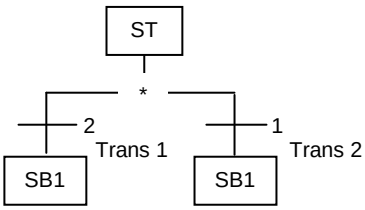
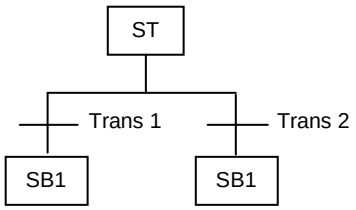
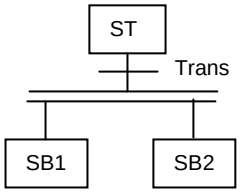
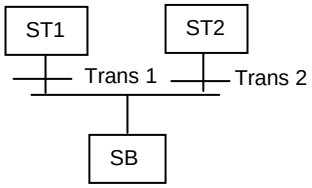
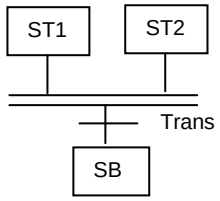
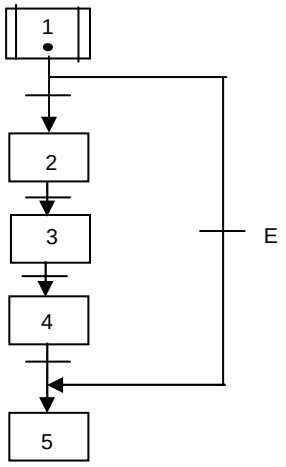
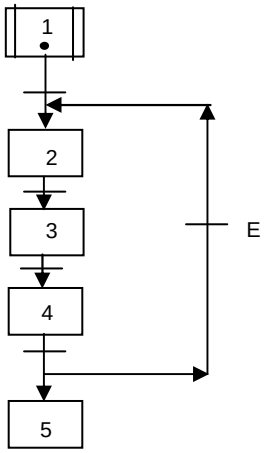
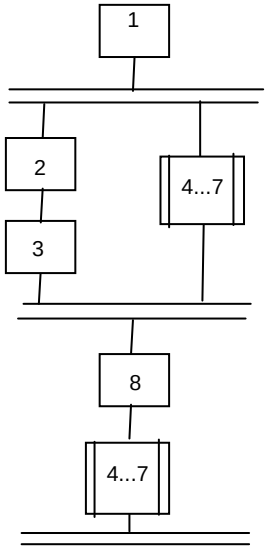
	<u>Sequência simples</u> • ST é desativado, tão logo, Trans=True (transição=verdadeira). • SB torna-se ativo assim que ST é desabilitado ou tornado inativo.
	<u>Caminho divergente</u> • Quando ST está ativo, as transições 1 e 2 são avaliadas segundo a ordem da direita para a esquerda. • Tão logo uma transição seja TRUE (Verdadeira), ST é colocado inativo e a SB subsequente é colocada ativa.
	<u>Caminho divergente com prioridade definida pelo usuário</u> • O usuário define a prioridade de precedência. O número menor tem a prioridade mais alta.
	<u>Caminho divergente sob o controle do usuário</u> • O usuário deve garantir que as duas ou mais transições sejam mutuamente exclusivas. • Obs.: Caso “Trans 1” e “Trans 2” venham a se tornar TRUE ao mesmo tempo, fica caracterizado o equivalente em Redes de Petri denominado “conflito confusão”.
	<u>Sequência simultânea</u> • Quando a “Trans” é TRUE, **todos** os passos subsequentes tornam-se ativos simultaneamente.
	<u>Convergência de sequência</u> • Quando um passo STn está ativo e sua “Trans n” sucessiva torna-se TRUE, o passo STn torna-se inativo e o passo SB é ativado.
	<u>Convergência de sequência simultânea</u> • Quando **todos** os passos STn estão ativos e **todas** as “Trans n” correspondentes tornam-se TRUE, os passos STn são desativados e o passo SB ativado.

Tabela 12 - Ilustração de três exemplos de sequência em SFC.

	Jump Condicional
	Repetição Condicional
	Sequência Repetitiva

É importante ressaltar que a condição de uma transição pode ser programada nas linguagens da 61131-3 IL, ST, LD e FBD.

3.3.2 Ações em SFC

A linguagem SFC permite representar uma série de ações básicas. A estrutura típica é: qualificador da ação, ação e indicador de variável. A figura 51 apresenta a representação gráfica de dois passos e suas respectivas ações. A tabela 13 apresenta as principais ações na linguagem SFC.

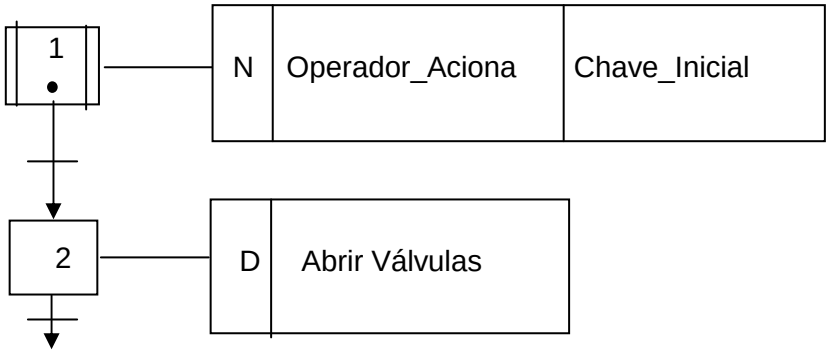


Figura 51 - Ilustração de dois passos e sãs respectivas ações.

Tabela 13 - Principais ações na linguagem SFC.

Qualificador	Descrição da ação
N	Ação simples. Executa a ação associada com o passo enquanto o mesmo está ativo. “Não memoriza”.
S	Set. Seta a ação ativa. “Memoriza”.
R	Reset. Reseta ou desativa a ação “setada” ou “memorizada”.
L	Ação por tempo limitado. Executa por um determinado tempo a ação.
D	Ação de tempo retardado. Executa continuamente uma determinada ação após um retardo (atraso) de tempo.
P	Ação pulsada. Executa uma única vez uma determinada ação.
SD	Ação de entrada com retardo prefixado.
SL	Ação “setada” com tempo limitado.

As figuras 52, 53, 54, 55, 56, 57, 58 ilustram exemplos de ações N, S, R, L, D, P e SD, SL e os seus respectivos diagramas de tempo.

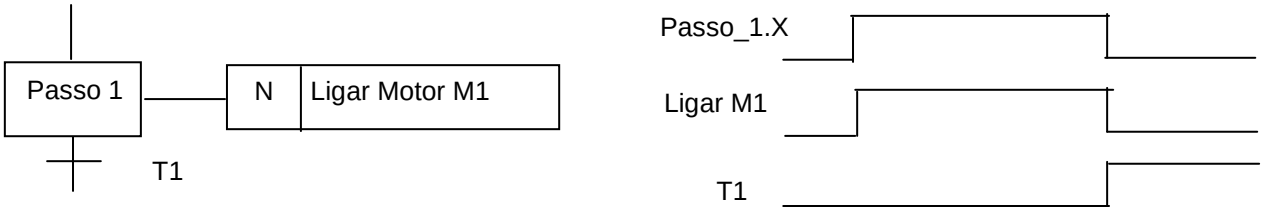


Figura 52 - Exemplo de Ação Simples (N).

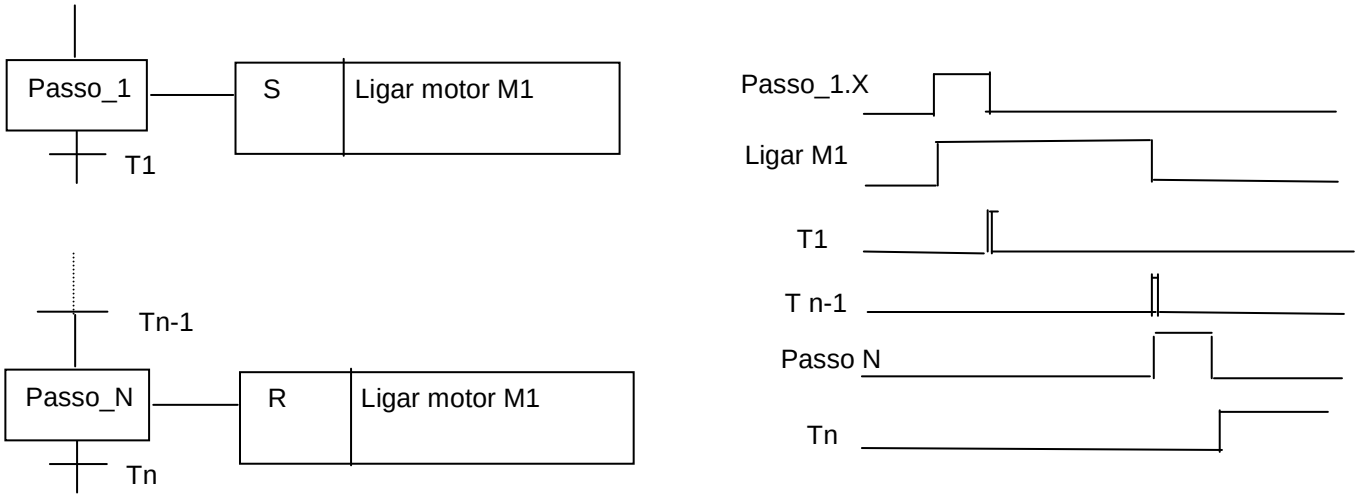


Figura 53 - Exemplo das ações: Ação Set (S) e Ação Reset (R).

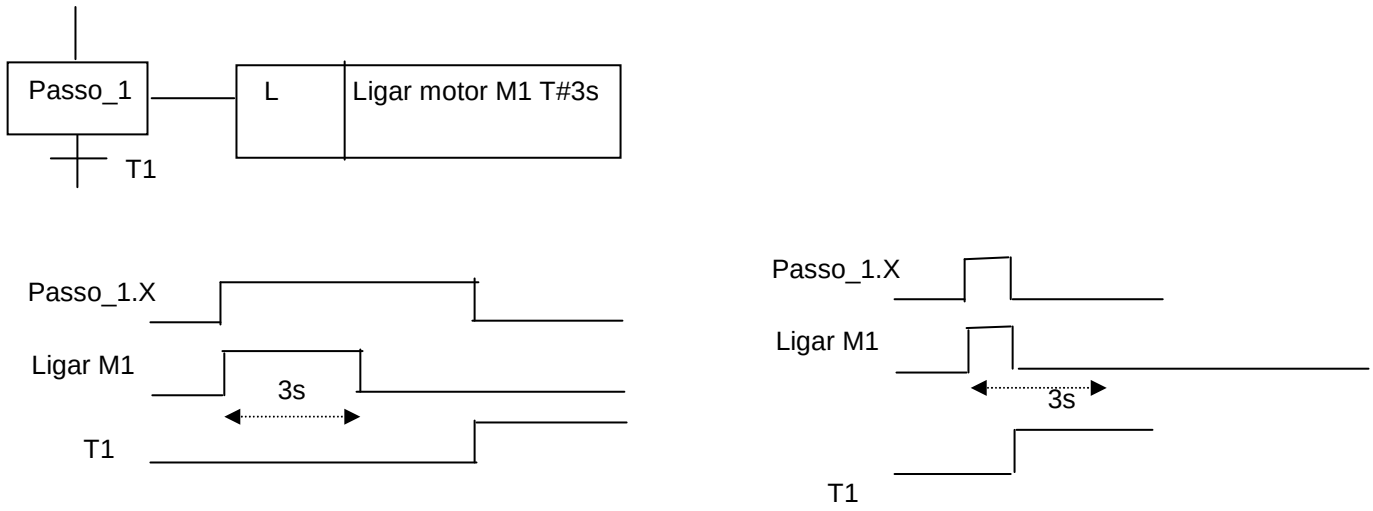


Figura 54 - Exemplo da Ação Tempo Limitado (L).

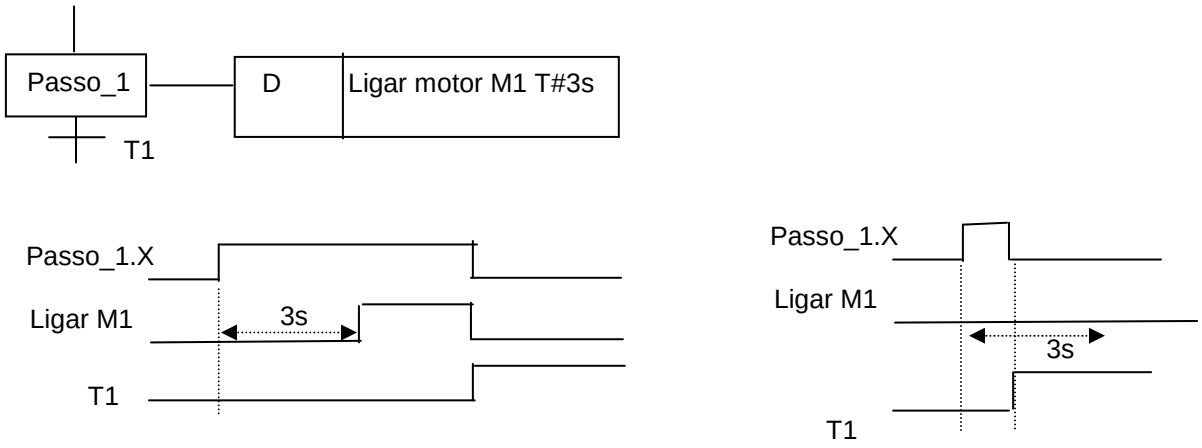


Figura 55 - Exemplo da Ação Tempo Retardado (D).

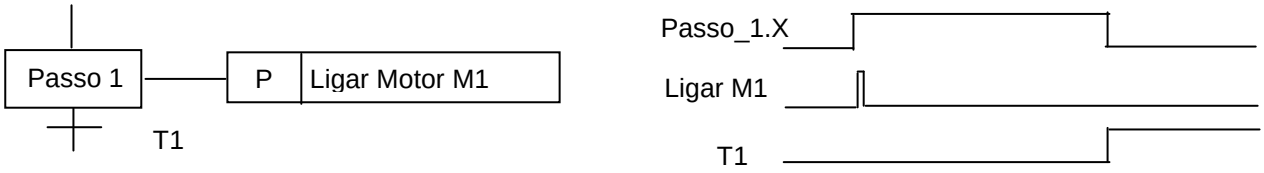
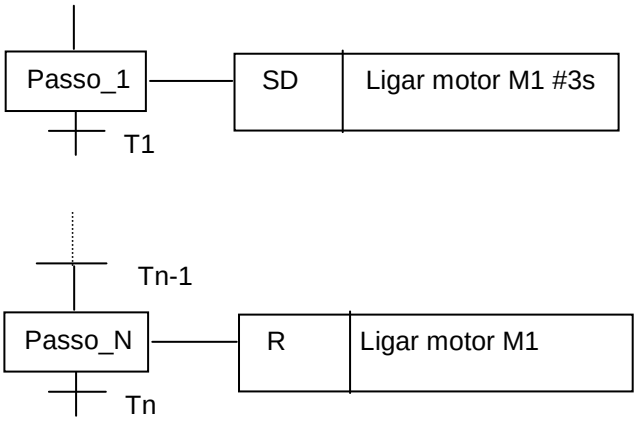


Figura 56 - Exemplo da Ação Pulsada (P).



Quando o passo 1 for ativado, a ação é memorizada e somente começará depois do tempo especificado. A Ação continuará até ser referendada no passo N com o Reset. Mesmo que o passo 1 seja desabilitado antes do início da ação a mesma ocorrerá caso o passo relativo ao Reset não esteja habilitado.

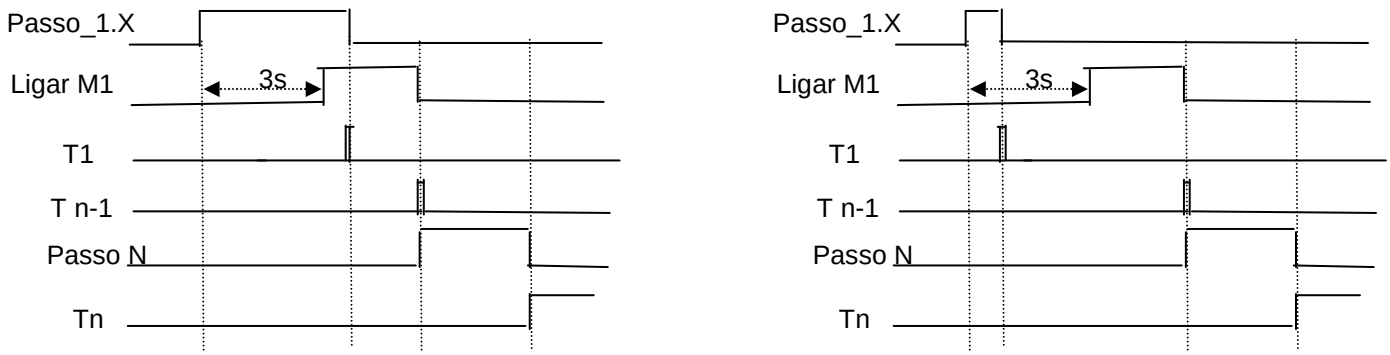


Figura 57 - Exemplo da Ação de Entrada com Tempo Prefixado (SD).

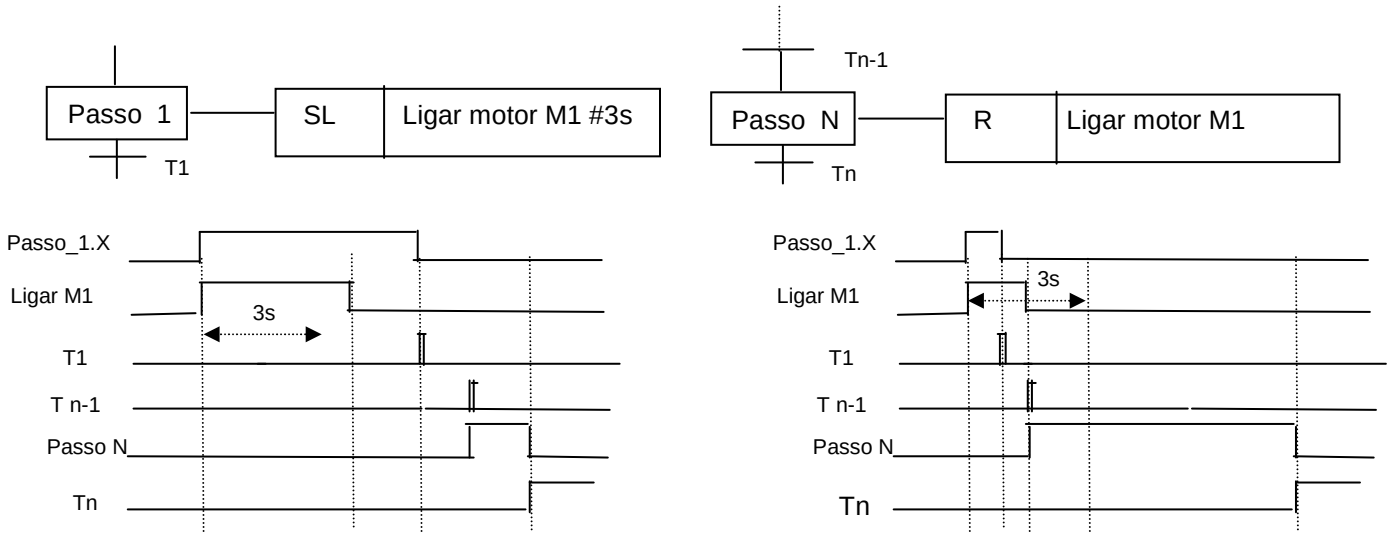


Figura 58 - Exemplo da Ação com Tempo Limitado (SL).

Suponha que um sistema de refrigeração seja composto por dois motores, M1 e M2. O motor M1 pode operar em duas velocidades: baixa (BV) e alta (AV). M1 opera em BV quando a chave de partida é acionada, e opera em AV quando um sensor de temperatura digital for acionado. Caso o sensor de temperatura seja desligado, o motor deverá operar em BV. O motor M2 é acionado pelo mesmo sensor de temperatura, caso um sensor de umidade digital indique umidade acima do normal. O motor M2 deve ser desligado depois de 60 minutos. A figura 59 apresenta o programa em SFC para executar o exemplo exposto.

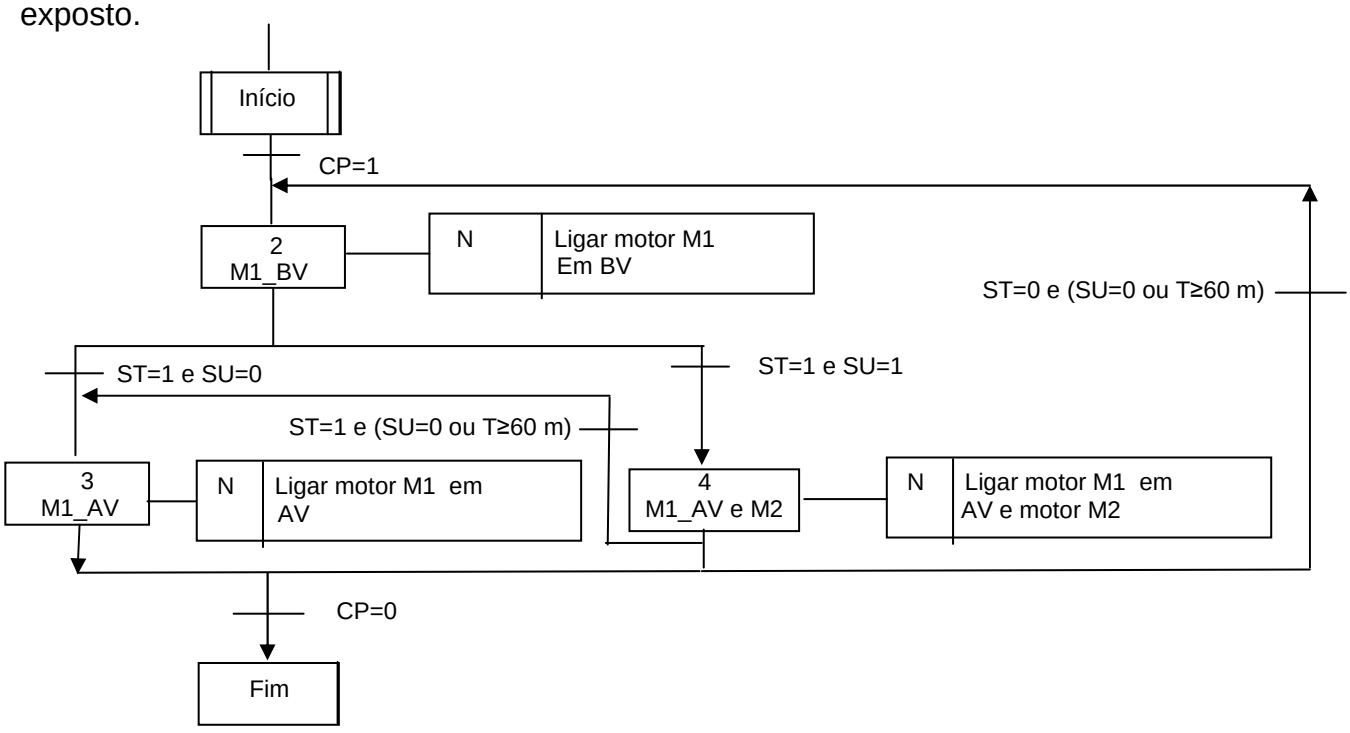


Figura 59 - Programa em SFC para executar o exemplo exposto.

A figura 60 apresenta um exemplo de trecho de programa extraído do manual "Logix5000 Controllers Sequential Function Charts".

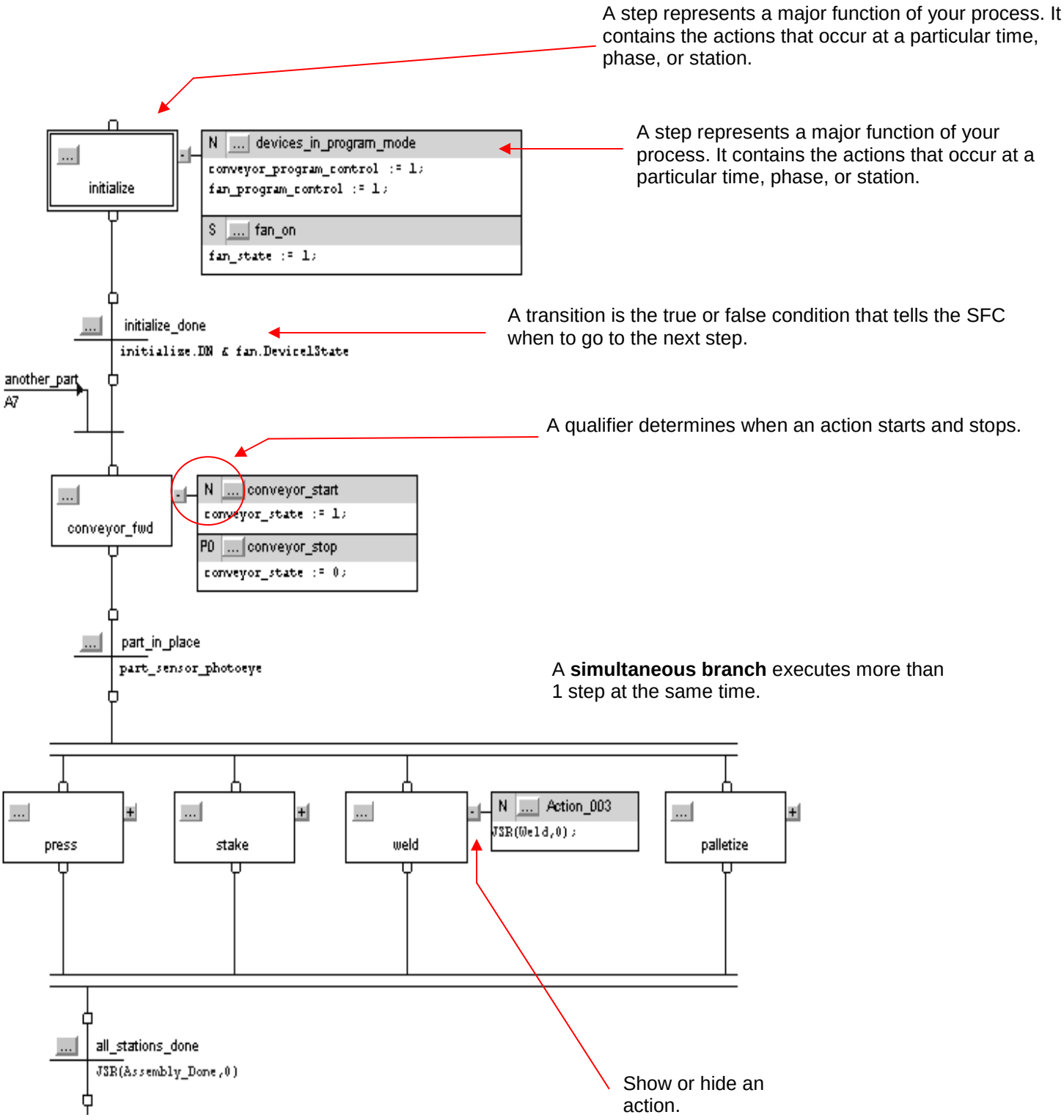


Figura 60 - Ilustrativo de exemplo de trecho de programa escrito em SFC no CLX. (Fonte: Rockweel Automation Logix5000 Controllers Sequential Function Charts Catalog Numbers 1756).

3.4 Linguagem FBD

O Diagrama de Blocos de Funções (*Function Blocks Diagram*) ou FBD é um conceito importante para compreensão e utilização dessa e das demais linguagens do padrão IEC 61131-3. Na linguagem FBD é definido o "POU" (*Program Organization Unit*), ou seja, a menor unidade de software de um programa, independente das demais, que corresponde a um bloco, que pode ser chamado com ou sem parâmetros e incluir outros POU em seu interior, conforme a necessidade. Uma POU pode ser programada em qualquer uma das linguagens existentes. Existem três tipos de POUs:

a) "Function" (FUN): É uma POU que pode ser atribuída com parâmetros não estáticos (sem memória), e que quando chamada com os mesmos valores de entrada retorna os mesmos resultados, como uma função matemática, por exemplo.

b) "Funtion Block" (FB): É uma POU que pode ser atribuída com parâmetros e que possui variáveis básicas internas (possui memória). Por exemplo: um contador ou um temporizador é uma "Function Block" FB.

c) "Program" (PROG): É a POU que representa o 'main program", ou seja, o programa principal de uma tarefa do CP.

A representação de uma POU na linguagem FBD, ou mesmo na linguagem LD, inclui partes comuns às linguagens textuais, tais como: parte principal, parte final, parte da declaração e parte do código. A parte da declaração pode ser tanto gráfica, como textual. A parte do código é dividida em redes que, por sua vez, são constituídas por: indicador de rede, comentário de rede e gráfico de rede.

A linguagem FBD é uma linguagem gráfica que descreve as relações entre sinais de entrada, parâmetros e sinais de saída, através de um fluxo algorítmico de dados entre esses elementos, através de blocos funcionais. Os elementos gráficos incluem caixas retangulares e declarações de controle de fluxo, conectadas por linhas horizontais e verticais. As entradas e saídas das caixas podem ser atribuídas a variáveis (parâmetros, variáveis temporárias), constantes, entradas ou saídas. As entradas e saídas também podem permanecer abertas (não conectadas), quando usualmente é assumida uma conectividade a um valor ou variável padrão.

A figura 61 ilustra um exemplo de um trecho de programa de CLP escrito de linguagem FBD.

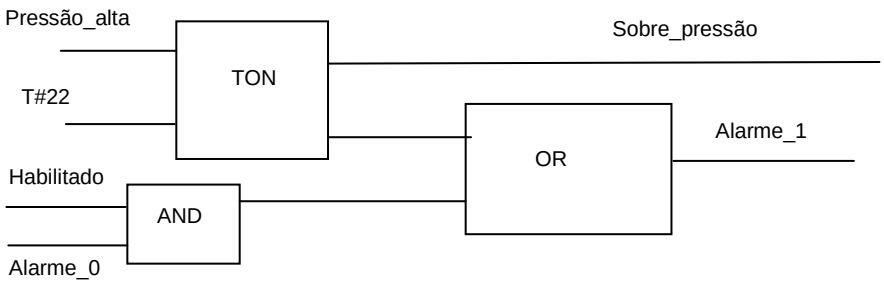


Figura 61 - Exemplo de um trecho de programa escrito em linguagem FBD.

Assim como na linguagem LD, na linguagem FBD uma POU é calculada de cima para baixo. Uma rede em FBD, para ser executada, deve ter cada elemento funcional calculado ou executado. Para cada elemento ser executado, esse deve ter todos os seus parâmetros de entrada definidos e ter todas as variáveis de entradas prontas e já calculadas, mesmo aquelas provenientes de outros blocos funcionais. O elemento é então executado e suas saídas calculadas. O processamento da rede FBD não estará finalizado enquanto todas as suas saídas não tiverem sido calculadas.

O grafo que conecta todas as entradas e saídas do FBD pode possuir loops, ou seja, uma entrada pode estar associada a uma saída do mesmo grafo FBD. Nesse caso, o fluxo de dados apresenta um caráter sequencial, ou seja, a função descrita depende de seus parâmetros usuais de entrada e também dos valores anteriores de suas próprias saídas.

No caso específico do CLX, praticamente todas as funções embutidas disponíveis na linguagem LD (timers, funções de lógica combinatória, funções aritméticas, etc.) também estão disponíveis como blocos funcionais ou funções para a linguagem FBD.

3.5 Linguagem ST

Texto estruturado (“*Structured Text*”) ou ST, como a linguagem IL, é uma linguagem de alto nível, fortemente textualizada, e que não emprega operadores de baixo nível como aqueles encontrados em uma linguagem de máquina do tipo “Assembly”. A linguagem oferece uma grande variedade de declarações abstratas e construções simplificadas, que

podem ser utilizadas em conjunto para descrever operações mais complexas, através de um processo algorítmico.

A linguagem se assemelha a outras linguagens populares, como o ANSI C e, principalmente, o PASCAL, da qual herda boa parte da semântica e dos elementos de programação, por exemplo, cada linha de um programa escrito em ST apresenta uma ou mais declarações separadas por ";" (sinalizador de término de linha). A linguagem ST também permite que os programas sejam estruturados, identados e comentados, conforme a necessidade. A tabela 14 ilustra as declarações na linguagem ST.

Tabela 14 - Declarações na linguagem ST (Karl and Michael Tiegelkamp)

Construção	Descrição	Exemplo	Explicação
:=	Declaração, associação ou atribuição	`D := 10;`	Atribui o valor da direita para o identificador à esquerda.

	Chamada de uma FB	`Nome_FB;`
		`Funcao1(10);`
		`Temp:=Funcao2;`
			Chama uma outra POU ou tipo de FUN ou FB, incluindo seus parâmetros.

RETURN	Retorna	`RETURN;`	Interrompe a POU atual e retorna.

IF	Seleção	`IF d<e THEN`
		`  f:=1;`
		`ELSE d=e THEN`
		`  f:=2;`
		`ELSE f:=3;`
		`END IF;`
			Construção típica IF THEN ELSE. No caso, seleciona alternativas por meio de expressões booleanas.

CASE	Seleção múltipla	`CASE f OF`
		`  1: g:=11;`
		`  2: g:=12`
		`ELSE`
		`ND CASE;`
			Seleção de blocos de declarações dependendo do valor da expressão.

FOR	Interação 1	`FOR h:=1 TO 10`
		`  BY 2 DO`
		`  F[h/2]:=h;`
		`END_FOR;`
			Loop iterativo, com índice "h", com condicionante de início e fim.

WHILE	Interação 2	`WHILE m>1 DO`
		`  N:=n/2;`
		`END WHILE;`
			Loop não iterativo, com condição para início e finalização.

REPEAT	Interação 3	`REPEAT`
		`i:=i*10;`
		`UNTIL i < 1000`
		`END REPEAT;`
			Loop não iterativo, com condição para finalização.

EXIT	Fim do loop	`EXIT;`	Terminações prematuras.
;	Declaração	`;  ;`	
//	Comentário de linha	`//Algoritmo 2`	Comentários de código
(* *)	Comentário de bloco	`(* Isso é` `  um teste *)`	Comentários de código

4 Acionamento de cargas e motores CA

O acionamento de motores de corrente alternada é uma das muitas funções que CPs devem executar em um processo automatizado. Este acionamento pode ser efetuado de forma convencional, empregando contatores, ou também de forma integrada, com *soft-starters* ou inversores de frequência. A figura 62 ilustra o diagrama macroscópico de blocos dos dois tipos de acionamento de motores, ilustrando apenas o circuito unifilar de potência.

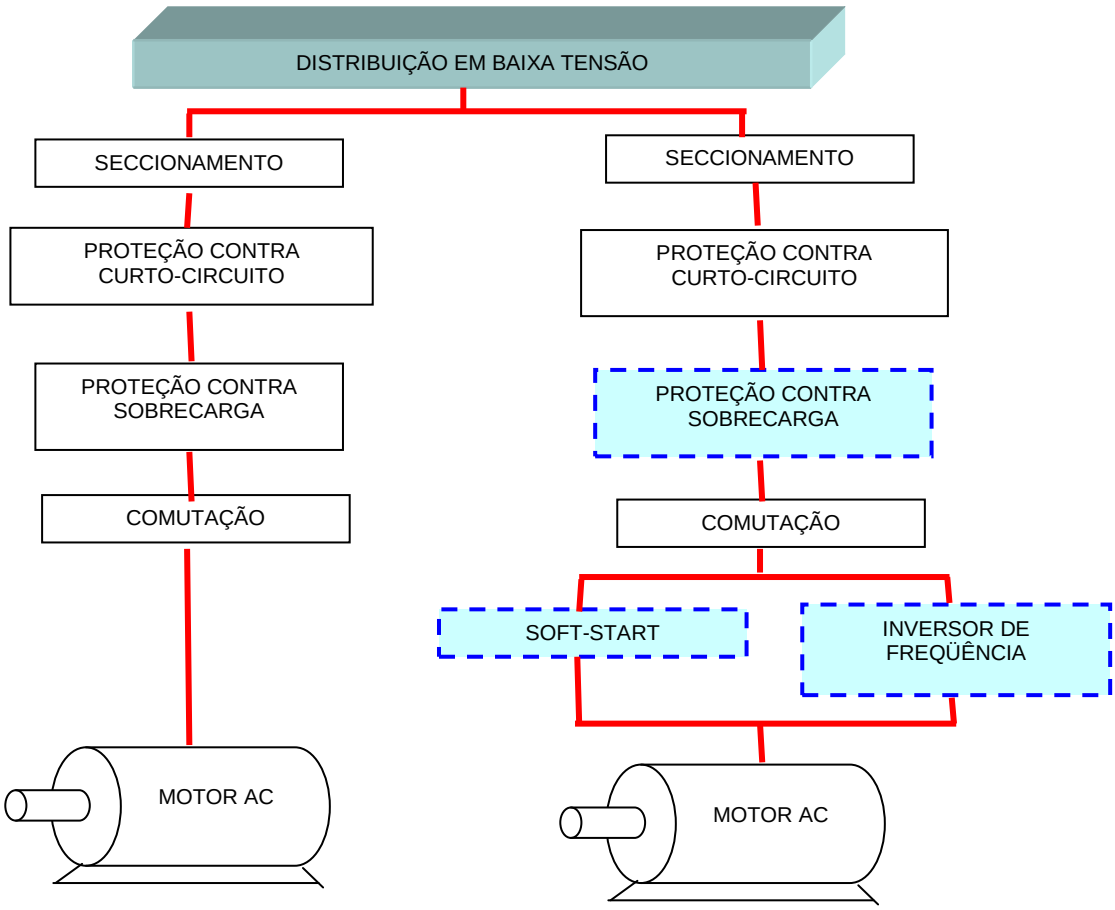


Figura 62 - Diagrama de blocos de acionamento (convencional e eletrônico) de motores trifásicos.

4.1 Resumo das normas para comutação e proteção (coordenação) de acionamento de motores AC e cargas elétricas

Os contatos dos contatores na comutação de cargas elétricas devem suportar os esforços originados pelas interrupções das correntes de alimentação. Diversos fatores interferem nesses esforços, tais como: frequências das operações de acionamento e

desligamento, valor das sobrecargas de tensão e de corrente, fator de potência da carga, tipo da carga, tipo de operação dos motores (partida, frenagem e inversão de rotação), etc. A vida útil dos contadores utilizados deve ser de 10^6 a 10^7 manobras. A tabela 15 apresenta um resumo da norma IEC 947-4 para o emprego de contatores.

Tabela 15 – Categorias de contatores (Fonte: Norma IEC 947-4 e Apostila Schneider Electric).

CATEG.	TIPO DE CARGA	USO DO CONTATOR	APLICAÇÕES TÍPICAS
AC1	Não indutiva (FP>0,9)	• Energização	Aquecimento, distribuição
AC2	Motores em anéis (FP>0,65)	• Partida • Desligar durante operação • Frenagem regenerativa	Trefiladoras
AC3	Motores de gaiola (FP 0,45 para I<100A) (FP 0,35 para I>100A)	• Partida • Desligar durante operação	Compressores, gruas, misturadores, bombas, escadas rolantes, ventiladores, transportadoras, ar condicionado
AC4	Motores de gaiola (FP 0,45 para I<100A) (FP 0,35 para I>100A)	• Partida • Desligar durante operação • Frenagem regenerativa • Inversão de sentido de rotação	Impressoras Trefiladoras

A coordenação das proteções é o ato de associar, de maneira seletiva, um dispositivo de proteção contra os curtos-circuitos com um contator e um dispositivo de proteção de sobrecarga. O objetivo é interromper a corrente de curto-circuito ou a corrente de sobrecarga, em tempo hábil, para proteger pessoas, equipamentos e a instalação elétrica. A tabela 16 apresenta os três tipos de coordenação definidos pela norma IEC.

Tabela 16 - Coordenação da proteção.

COORDENAÇÃO TIPO 1	É permitida uma degradação do contator e do relé térmico sob duas condições: • Nenhum risco ao operador. • Todos os demais equipamentos da devem estar protegidos.
COORDENAÇÃO TIPO 2	É permitido que haja uma probabilidade da soldagem dos contatos dos contatos do contator, ou do relé térmico, desde que ambos possam ser reparados facilmente. Obs.: Valem as mesmas duas condições da coordenação tipo 1.
COORDENAÇÃO TOTAL	Não é aceito nenhum dano ou alteração de ajuste operacional da proteção, assim como valem as mesmas condições da coordenação tipo 1.

4.2 Exemplo: Automação de uma esteira transportadora

Suponha uma determinada esteira transportadora que possua os seguintes dispositivos e componentes, como ilustrados na figura 63.

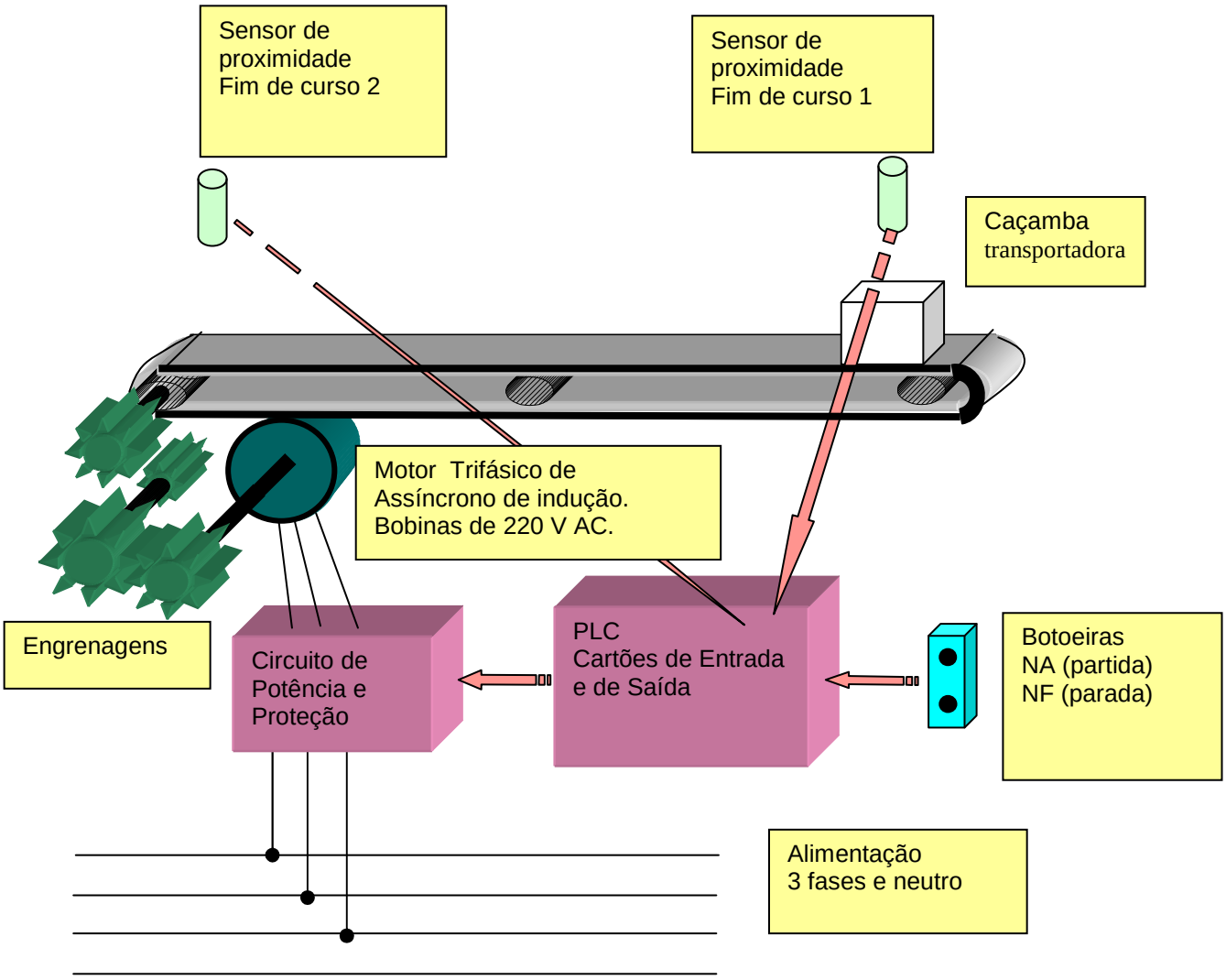


Figura 63 - Representação do sistema de automação e da esteira transportadora.

4.2.1 Requisitos e algoritmo do processo de transporte

a) A botoeira de partida é do tipo “push-button”, normalmente aberta, e quando acionada, coloca o sistema em operação para realizar um ciclo completo, ou então terminar um ciclo interrompido pelo acionamento do botão de emergência.

b) O motor deve ser acionado (ou seja, deve ser partir) com as bobinas ligadas em estrela e, após cinco segundos, a ligação das bobinas do estator deve ser automaticamente comutada para a ligação triângulo.

c) O motor irá acionar a esteira até que a caçamba de transporte de material atinja a posição 2 (esta situação é detectada pela chave fim de curso 2). Nessa ocasião, o motor deve ser desligado por 15 segundos (tempo necessário para simular a carga/descarga do material) e, em seguida, a sua rotação deve ser automaticamente revertida de modo que a caçamba retome para a posição 1.

d) A botoeira de parada (normal ou “emergência”), do tipo normalmente fechada, deve interromper o movimento da esteira em qualquer etapa do processo. Ao ser novamente acionada a botoeira de partida, o processo deve ser completado até o retorno da caçamba à posição 1.

e) Em qualquer etapa do processo, a partida do motor deve ser feita sempre na ligação estrela e revertida para triângulo após cinco segundos de funcionamento.

4.2.2 Descrição do circuito de potência

O circuito de potência necessário para o controle do motor utiliza 4 contatores trifásicos. Dois desses contatores (C e D) são utilizados para fazer as ligações estrela e triângulo, sendo que os outros dois (A e B) executam o controle do sentido de rotação do motor.

O contator A é responsável por girar o motor no sentido de levar a caçamba da posição 1 para 2. O contator B, por sua vez, inverte o sentido de rotação por meio de uma mudança na seqüência de fase do trifásico.

O contator C, quando energizado, liga as bobinas do motor em estrela; o contator D altera as ligações das bobinas para a ligação para triângulo (delta).

Os contatos auxiliares normalmente fechados dos contatores que estão ligados em série com as bobinas dos contatores A, B, C e D proporcionam um intertravamento físico (não somente no diagrama LD do PLC) entre os pares de contatores A,B e C,D. Esse intertravamento de *hardware* é necessário por razões de segurança, uma vez que o acionamento simultâneo dos contatores A e B ou C e D provoca um curto circuito na alimentação trifásica. As figuras 64 e 65 ilustram as ligações descritas.

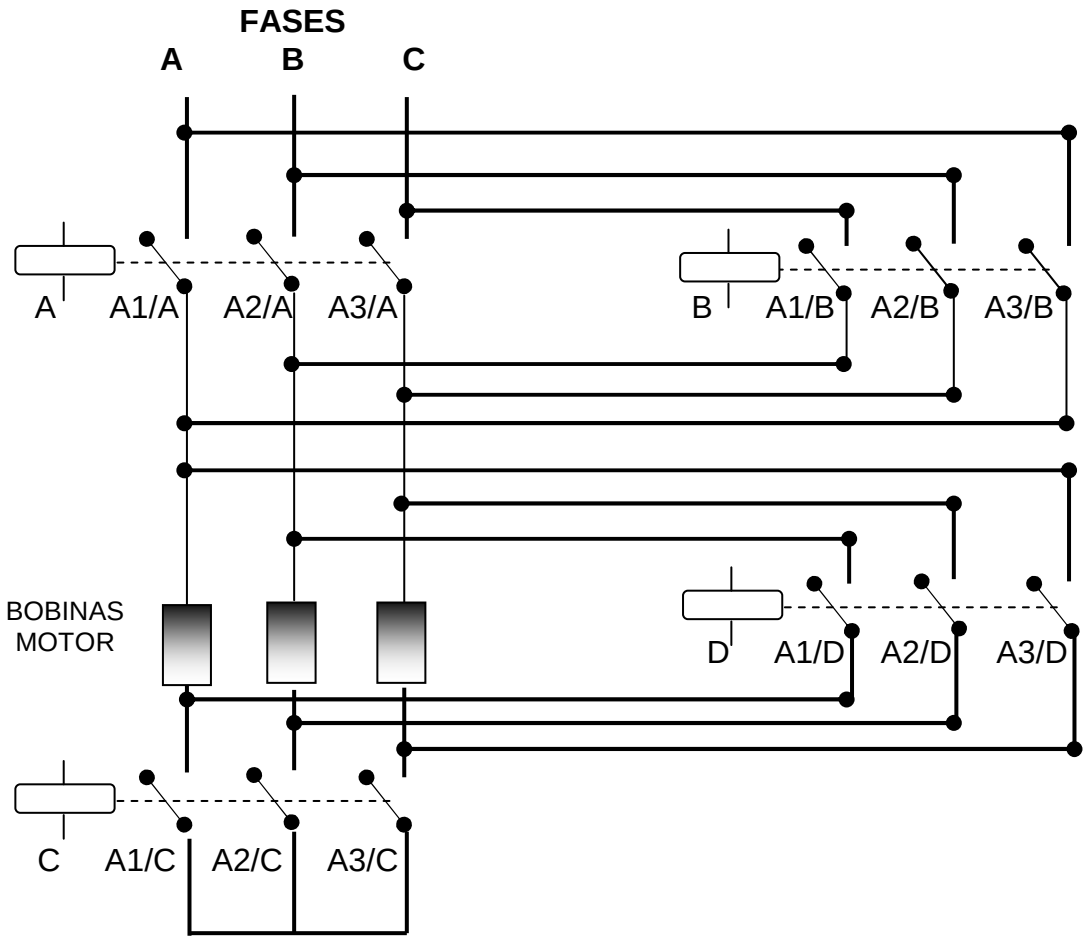


Figura 64 - Reversão estrela/triângulo e sentido de rotação.

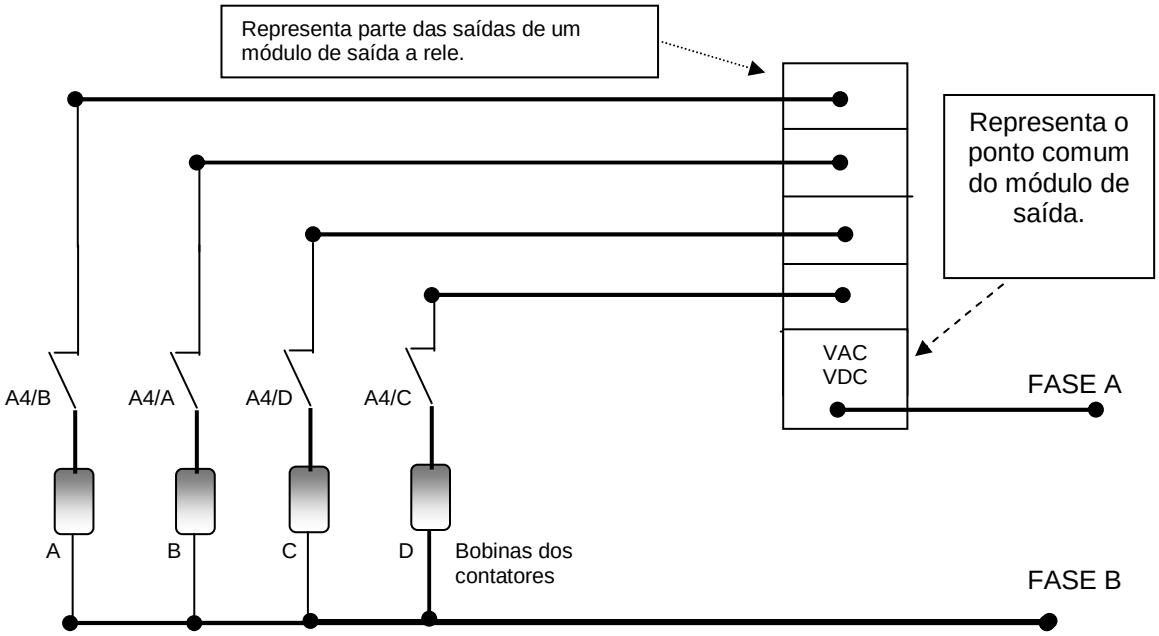


Figura 66 - Ligação das bobinas com intertravamento de *hardware* por meio dos contatos auxiliares dos contatores.

A tabela 17 apresenta uma sugestão para a utilização das entradas do CP CLX da Rockwell Automation conforme as indicações das figuras 64 e 65.

Tabela 17 – Sugestão de ligações das figuras 64 e 65 utilizando CPs da família SLC da Rockwell Automation.

Variável	saída	Endereço
Bobina A	8	O:2/8
Bobina B	9	O:2/9
Bobina C	10	O:2/10
Bobina D	11	O:2/11

Variável	Tipo	entrada	Endereço
P.B. partida	NA	3	I:1/3
P.B. parada	NF	1	I:1/1
fim de curso 1	NA	6	I:1/6
fim de curso 2	NA	7	I:1/7

4.2.3 Descrição do programa escrito na linguagem LD

A tabela 18 apresenta a descrição do programa proposto, mostrado em seu diagrama LD na Figura 67.

Tabela 18 – Descrição e explicação do programa em LD.

Linha	Descritivo do programa
1	Conforme pode ser observado na Linha 1 a variável sentido indica a direção no deslocamento da caçamba. Quando a variável sentido é falsa, a Caçamba desloca-se no sentido da posição 1 para 2, e quando a variável sentido é verdadeira ocorre o retomo para a posição 1. Ao se energizar o PLC, com a caçamba na posição inicial, a variável sentido torna-se falsa. Ao se atingir a posição 2, devido ao fechamento da chave fim de curso 2, a variável sentido toma-se verdadeira indicando o retomo da caçamba. Quando a caçamba retoma para a posição 1, a variável retornou toma-se verdadeira o que acarreta em novamente zerar a variável sentido.
2	Na Linha 2 utiliza-se a instrução OSR, a qual torna a linha verdadeira, durante uma varredura, quando ocorre uma transição de falsa para verdadeira na entrada da instrução. Dessa forma, quando a caçamba chega à posição 1 a variável retornou toma-se verdadeira durante um ciclo de scan do programa. Isto é suficiente para zerar novamente a variável sentido preparando desta forma o próximo cicio.
3	Na Linhas 3 e 4 as variáveis vai e volta controlam as saídas digitais que energizam as bobinas dos contatores A e B, respectivamente. Quando a variável sentido é falsa a variável vai pode ser habilitada, caso contrário, a habilitação ocorre para a variável volta.
4	A variável Timer 2/DN na linha 5 inicia a volta da caçamba quando o timer 2 termina a temporização referente ao tempo de carga/descarga na posição 2.
5	
6	O temporizador da linha 7 determina no tempo que a caçamba deve ficar parada na posição fim de curso 2.
7	As instruções da Linha 7 contam o tempo necessário para o chaveamento da ligação estrela para delta.
8	As Linhas 8 e 9 por sua vez, controlam as saídas digitais que energizam as bobinas dos contatores C e D (ligação estrela e delta do motor, respectivamente). O Bit timer 1/TT fica no estado 1 quanto o respectivo temporizador estiver temporizando sem que o valor do acumulado ter atingido o valor do pré-selecionado.
9	

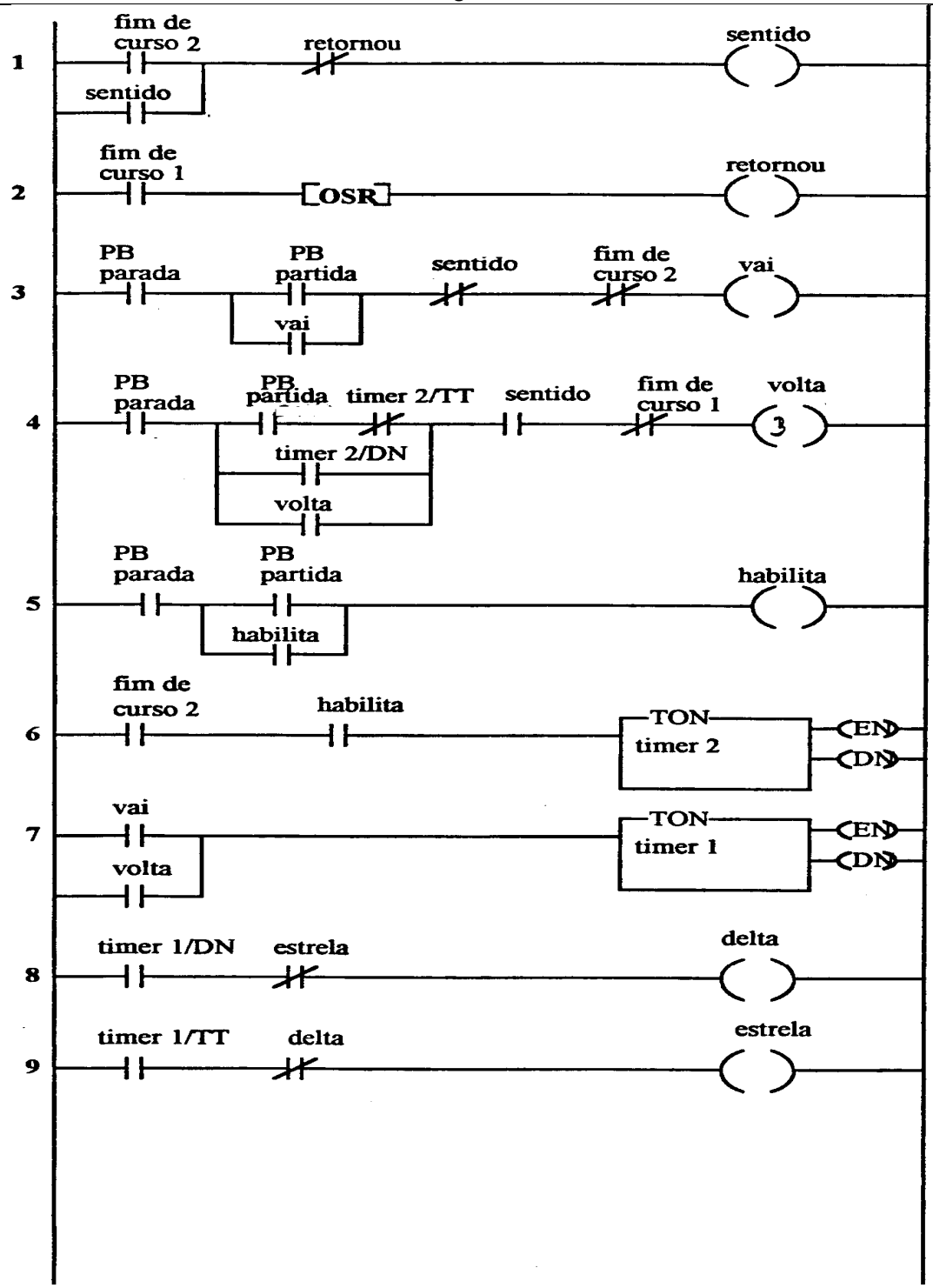


Figura 67 - Diagrama em LD genérico para implementar o algoritmo do processo especificado.

Fonte: (Moraes, Senger, Pereira Apostila CLP 1995).

A figura 68 ilustra o diagrama em LD específico para a plataforma SLC 500 da Rockwell, para implementar o algoritmo descrito, considerando a arquitetura de hardware disponível e também as entradas definidas na tabela de endereçamento proposta.

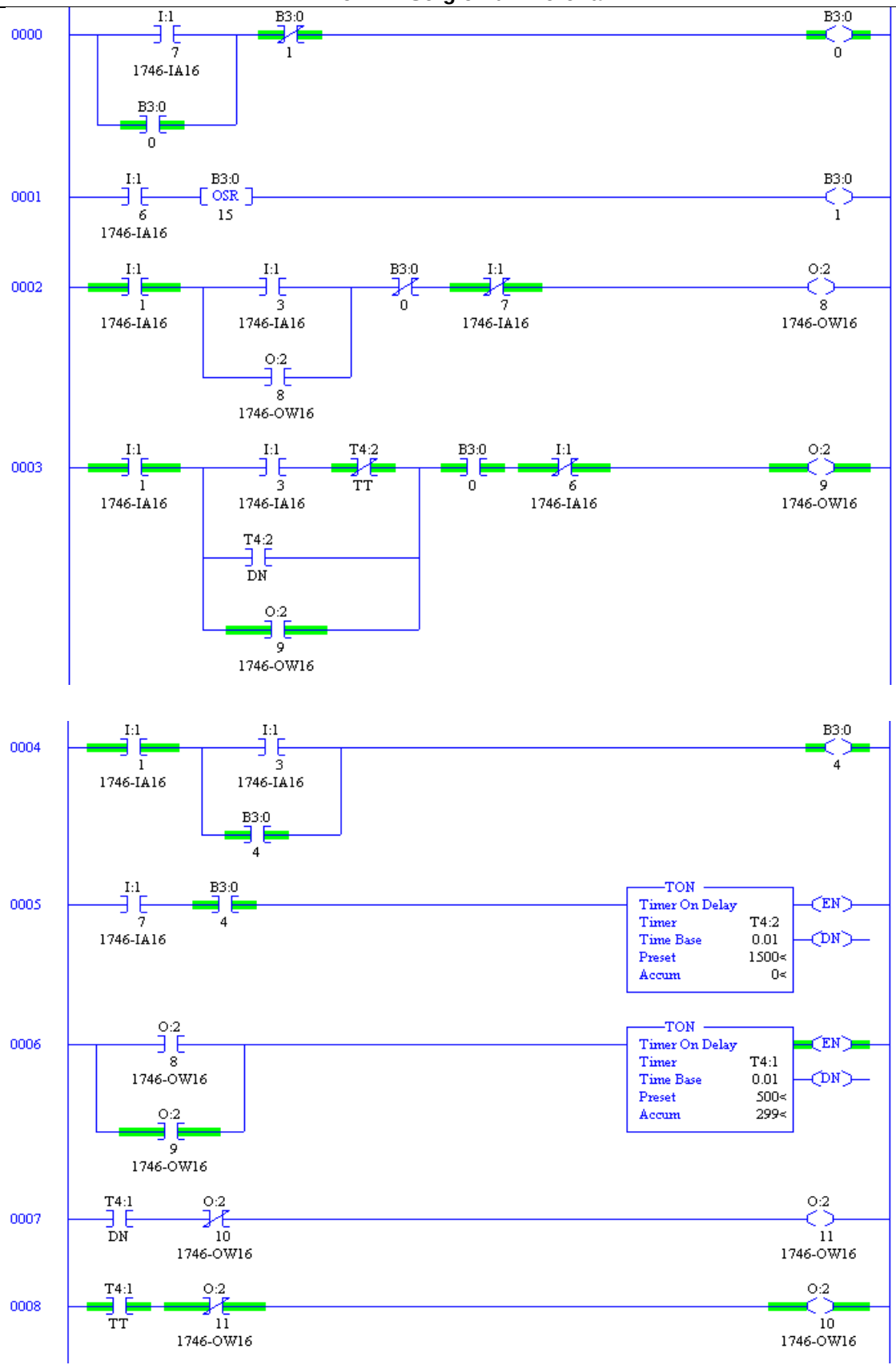


Figura 68 - Diagrama LD com o CLP SLC 500 para implementar o algoritmo do processo.

4.3 Dispositivos eletrônicos para partida de motores

4.3.1 Soft-starters

Podem ser empregadas tanto em motores de CC como em motores CA. Quando o motor é acionado, a tensão é aumentada progressivamente de forma temporizada ou acompanhando a evolução da corrente do motor. A elevação da tensão pode ser programada e alterada conforme a curva de aceleração desejada para o equipamento.

A operação de um *soft-starter* em CA é baseada no emprego de SCRs (tiristores) em uma ponte tiristorizada, disparada por um circuito de comando, de forma a variar o nível de tensão da saída. Um *soft-starter* é composto de: um circuito de potência, um circuito de comando e uma interface IHM. A figura 69 apresenta um diagrama de blocos de um *Soft-Starter* genérico. As principais funções são:

a) Controle das rampas de aceleração de desaceleração do nível de tensão.

b) Limitação de corrente ajustável.

c) Conjugado de partida.

d) Frenagem por injeção de corrente contínua.

e) Proteção por acionamento de sobre carga.

f) Proteção do motor contra aquecimento devido à sobrecarga.

g) Detecção de desequilíbrio ou falta de fase.

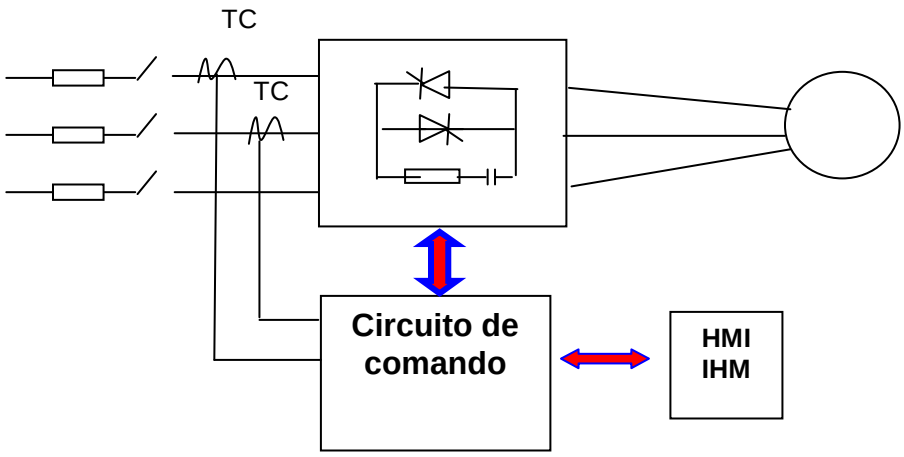


Figura 69 - Diagrama de blocos de um *soft-starter* genérico.

A curva da tensão ao longo do tempo de um *soft-starter* pode ser regulada conforme a necessidade. A figura 70 apresenta uma curva genérica para uma rampa de aceleração da tensão ao longo do tempo. A figura 71 apresenta a forma de onda aplicada no motor para que o mesmo tenha o módulo da tensão instantânea como ilustrado na figura 70. A figura 72 apresenta uma curva genérica para a rampa de desaceleração da tensão, e a figura 73 apresenta a forma de onda da tensão correspondente.

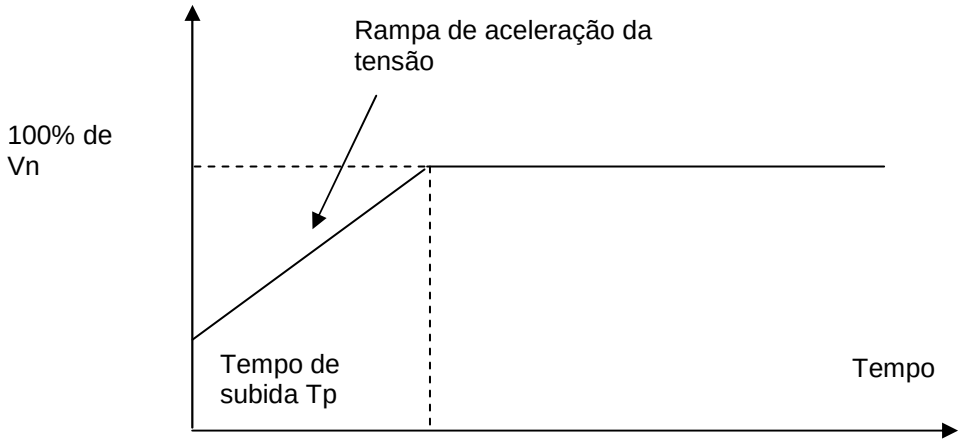


Figura 70 - Rampa de aceleração de subida da tensão no motor.

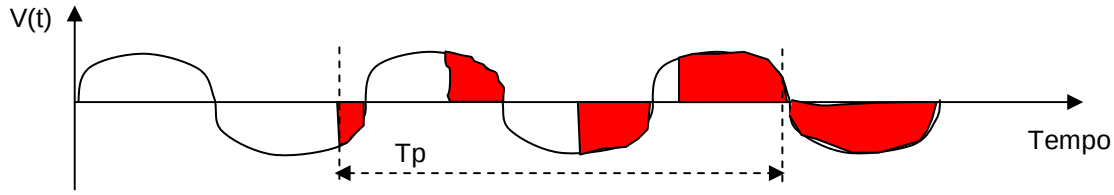


Figura 71 - Forma de onda da tensão aplicada no motor para que a tensão tenha a curva de aceleração ilustrada na figura 70.

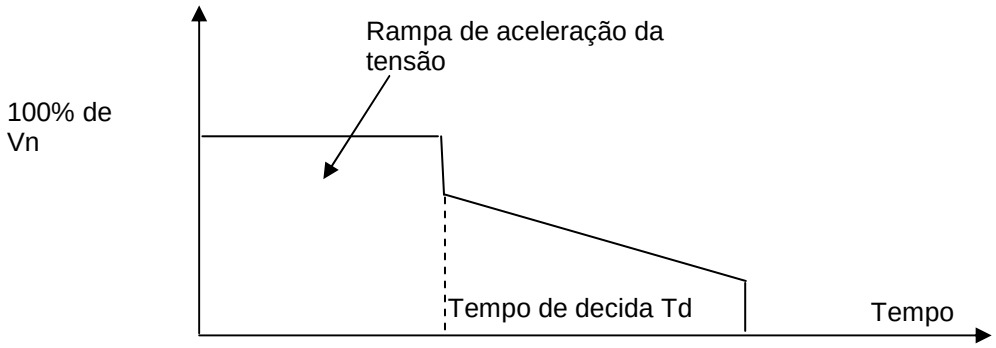


Figura 72 - Rampa de desaceleração da tensão no motor.

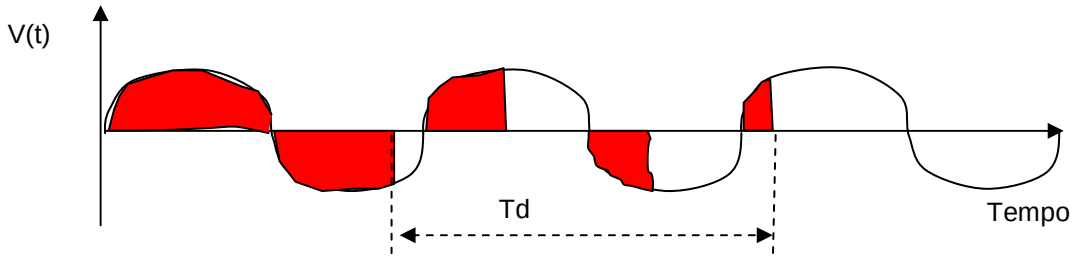


Figura 73 - Forma de onda da tensão aplicada no motor para que a tensão tenha a curva de desaceleração ilustrada na figura 72.

Os *soft-starters* podem ser ligados de diversas maneiras: Ligação direta, Ligação com contator em paralelo (*bypass*), ligação para acionamento sequencial de diversos motores e ligação para acionamento simultâneo de diversos motores. A figura 74 ilustra o esquema de ligação direta.

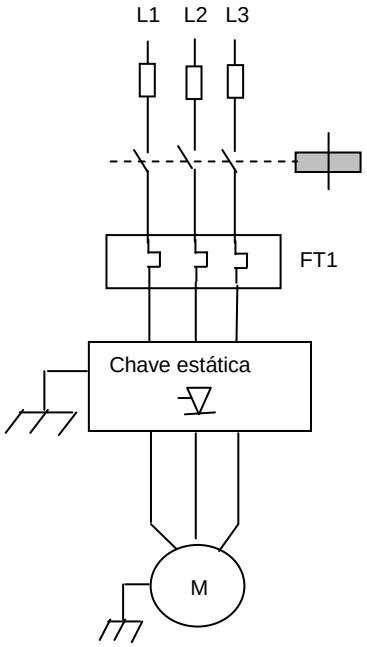


Figura 74 - Esquema de ligação direta com um *soft-starter*.

4.3.2 Inversores de frequência

Assim como os *soft-starters*, os inversores de frequência são equipamentos eletrônicos desenvolvidos para variar a velocidade de motores de indução trifásicos. Entretanto, a complexidade e a possibilidade de controle de velocidade de motores trifásicos dos inversores de frequência superam significativamente os *soft-starters*. A figura ilustra o diagrama de blocos genérico de um inversor de frequência. A figura 75 ilustra um diagrama de blocos de um inversor de frequência genérico. A figura 76 ilustra parcialmente os principais circuitos, e as respectivas formas de onda de tensão de um inversor de frequência.

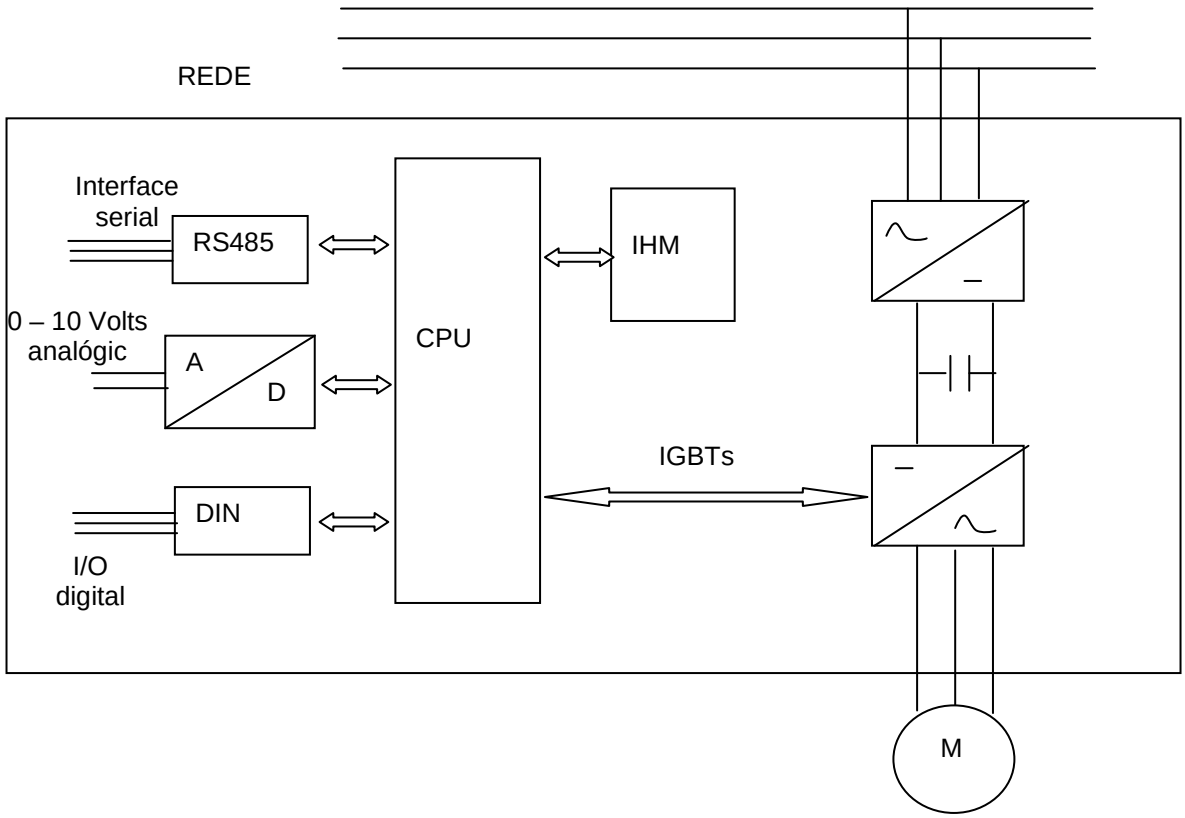


Figura 75 - Diagrama de blocos de um inversor de frequência.

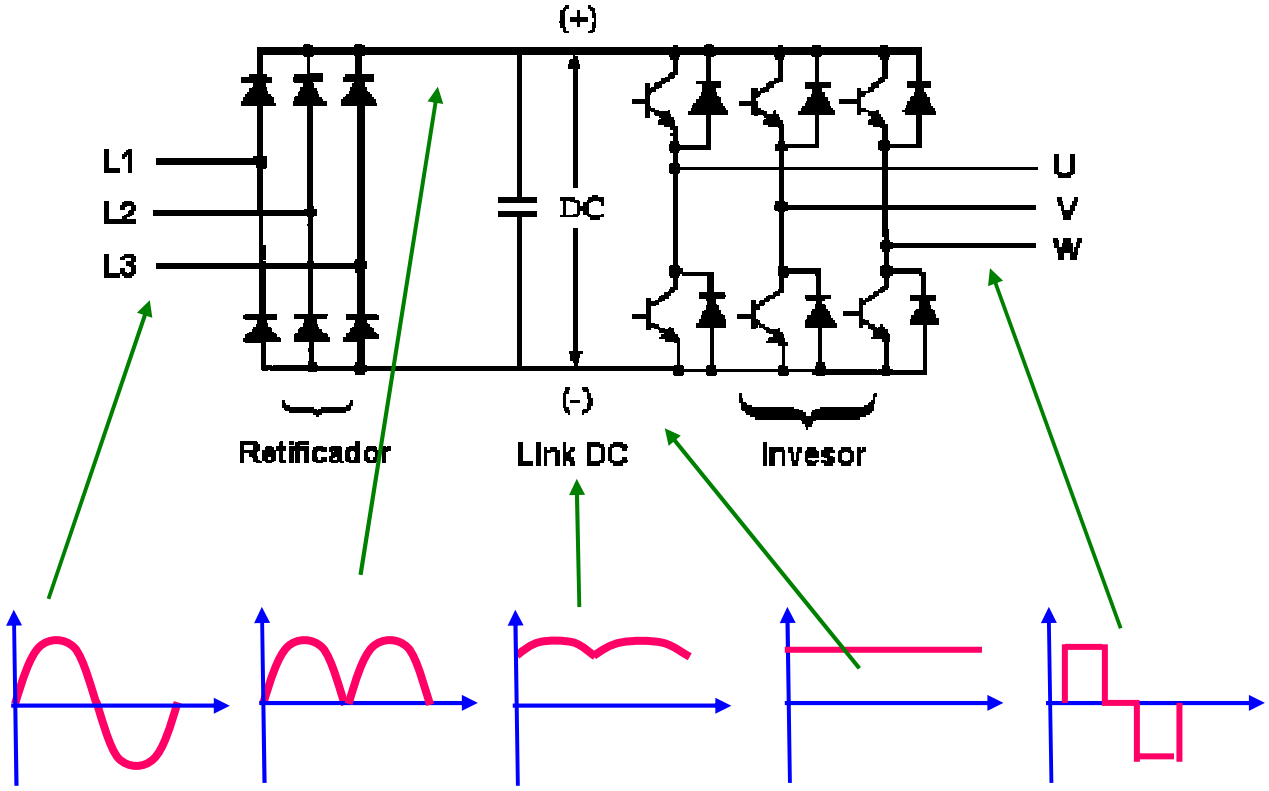


Figura 76 - Ilustrativo parcial dos circuitos e das respectivas formas de onda de tensão de um inversor de frequência.

O nome correto para um inversor de frequência deveria ser conversor de frequência. Entretanto, o nome popularizado foi o de inversor. Também é comum encontrar alguns modelos comerciais que possuem um filtro no link CC para fornecer uma tensão senoidal na saída. A sigla IGBT significa *Transistor bipolar de porta isolada*. A figura 77 ilustra um inversor contendo o filtro no link CC.

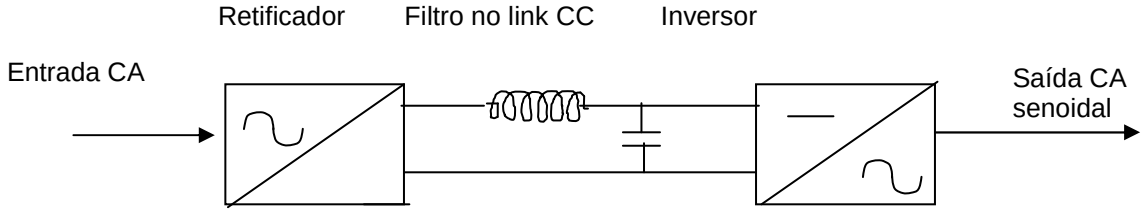


Figura 77 - Inversor de frequência com filtro no link CC.

4.3.3 Explicação e análise de um inversor monofásico

O princípio de operação dos inversores de frequência baseia-se no chaveamento e comutação da tensão/corrente por meio de transistores. A figura 78 ilustra o circuito básico de um inversor monofásico. A figura 79 ilustra a operação do mesmo e a figura 80 ilustra a forma de onda de tensão obtida na saída em função do chaveamento dos transistores.

Observa-se que durante a operação, os transistores T1 e T4 são disparados (ligados), enquanto que os transistores T2 e T3 operam como um circuito aberto. Quando isto ocorre, a corrente circula pelo motor no sentido A para B. Por outro lado, quando os Transistores T2 e T3 são disparados, os transistores T1 e T4 são desligados e funcionam como circuitos abertos. Neste caso a corrente pelo motor circula no sentido B para A.

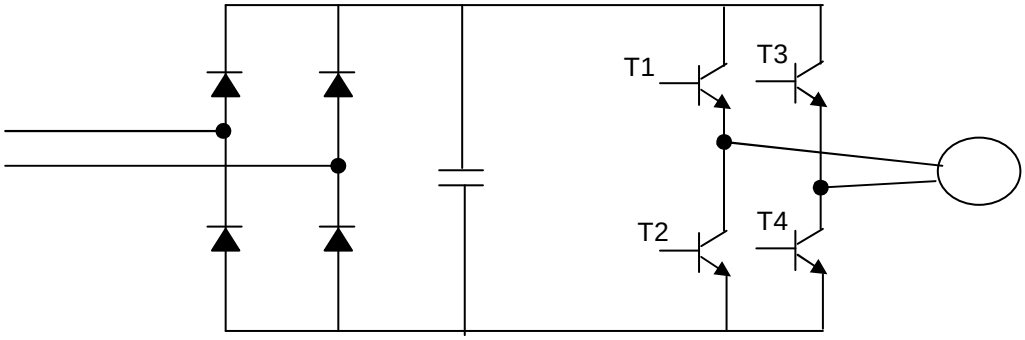


Figura 78 - Exemplo de um inversor monofásico.

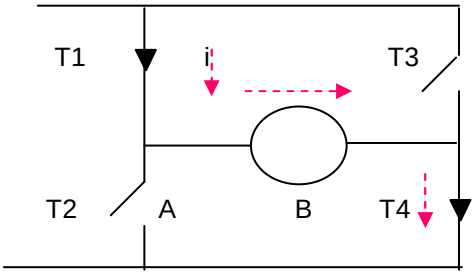


Figura 79 - Ilustrativos da operação de um Inversor monofásico comutando transistores T1 e T2.

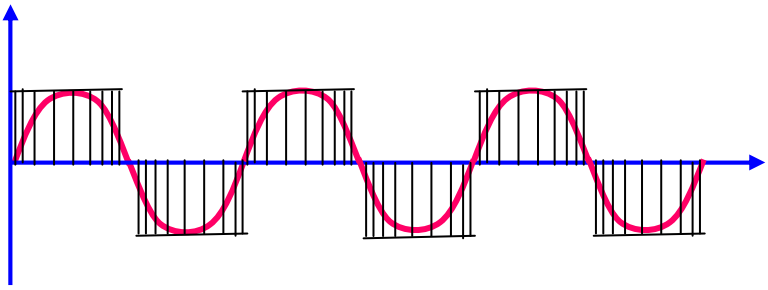


Figura 80 - Forma de onda gerada devido ao chaveamento.

4.3.4 Classificação de inversores de frequência

Existem dois tipos de controle aplicado em inversores de frequência: controle escalar e controle vetorial.

Controle escalar: Também chamado de controle Volt/Herz, objetiva manter a relação tensão *vs.* frequência constante. A figura 81 apresenta a curva genérica típica de um inversor de frequência que emprega controle escalar.

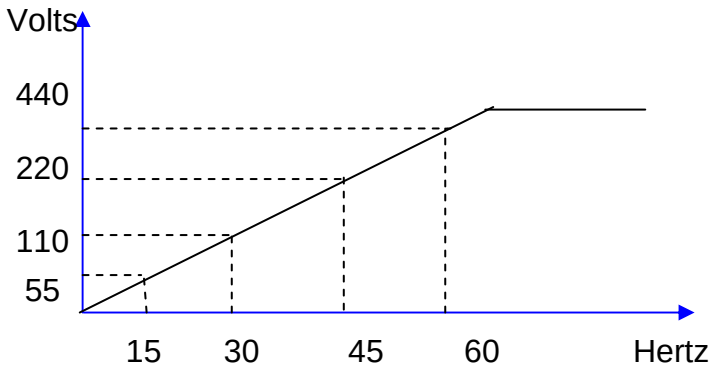


Figura 81 - Tensão/frequência de um inversor escalar.

O controle escalar aplicado a um inversor de frequência além de possuir um custo menor em relação ao controle vetorial possui as seguintes características:

a) O controle escalar é utilizado em aplicações normais, que não requerem elevada dinâmica (grandes acelerações e frenagens), elevada precisão, nem controle de torque.

b) Possui precisão de até 0,5% da rotação nominal para sistemas sem variação de carga, e de 3% a 5% com variação de carga de 0 a 100% do torque nominal.

c) Normalmente esse tipo de controle é realizado em malha aberta, ou seja, sem a leitura da velocidade do motor através de um sensor, e a precisão da velocidade é função do escorregamento do motor, que varia em função da carga.

d) Esse controle não é indicado para motores que rodam a baixas velocidades (abaixo de 5 Hz), pois o torque em baixas velocidades é geralmente pequeno porque a queda de tensão afeta significativamente a magnitude da corrente de produção de fluxo. Muitos inversores de frequência incluem um torque extra de partida, que permite que a relação V/F seja aumentada na partida para aumentar o fluxo e, consequentemente, o torque de partida.

Controle Vetorial: A corrente que circula em um estator de um motor de indução pode ser dividida em duas: *Im* que é a corrente de magnetização que produz o fluxo, e a *Ir* que é a corrente produtora de torque. Assim sendo, a corrente total é a soma vetorial das duas correntes, e o torque produzido no motor é proporcional ao produto vetorial das duas componentes da corrente.

A estratégia do controle vetorial é calcular a corrente de cada um dos vetores e possibilitar a separação do controle da corrente de fluxo e/ou o controle da corrente de torque, em todas as condições de velocidade e de torque. Seu objetivo é manter o fluxo constante de corrente no motor.

O cálculo desses vetores envolve a medição das variáveis disponíveis do estator, tais como tensão de fase, relação de fase, frequência, além da velocidade do eixo, etc. Além disso, è necessário conhecer ou calcular (estimar) os seguintes parâmetros do motor: resistência do estator, resistência do rotor, indutância do estator, indutância do rotor, indutância de magnetização e a curva de saturação.

O inversor de frequência vetorial, além de controlar a tensão e a frequência aplicadas ao motor, controla também a corrente rotórica, e como o torque depende diretamente desta corrente, o controle vetorial permite um controle de velocidade e de torque muito eficiente.

Esse tipo de controle apresenta melhorias se comparado com a técnica V/Hz através do provimento de ambos, magnitude e ângulo entre tensão e corrente, em vez do V/Hz que controlava apenas a magnitude. O ângulo de tensão do motor controla a quantidade de corrente que vai para o fluxo do motor, habilitado pelo estimador corrente de torque. Por meio do controle desse ângulo, a operação em baixas velocidades e o controle de torque é substancialmente melhorado com relação ao V/Hz.

O controle vetorial fornece as seguintes vantagens em relação ao controle escalar:

a) Elevada precisão de regulação de velocidade;

b) Alta performance dinâmica;

c) Controle de toque linear para aplicações de posição ou de tração;

d) Operação suave em baixa velocidade e sem oscilações de torque, mesmo com variação de carga.

Os valores típicos para esses inversores são:

a) Regulação de velocidade: 0,1%.

b) Regulação de torque: não tem.

c) Torque de partida: 250%.

d) Torque máximo (não contínuo): 250%.

e) Controle em malha aberta.

f) O inversor já conhece os parâmetros da máquina pela auto-sintonia.

g) Possui melhor desempenho se comparado a um controle escalar (V/F).

A figura 82 apresenta a curva genérica de corrente e de torque de um motor trifásico de indução, de dois pares de pólos, ligado em 60 Hz, acionado por um inversor de frequência de controle vetorial.

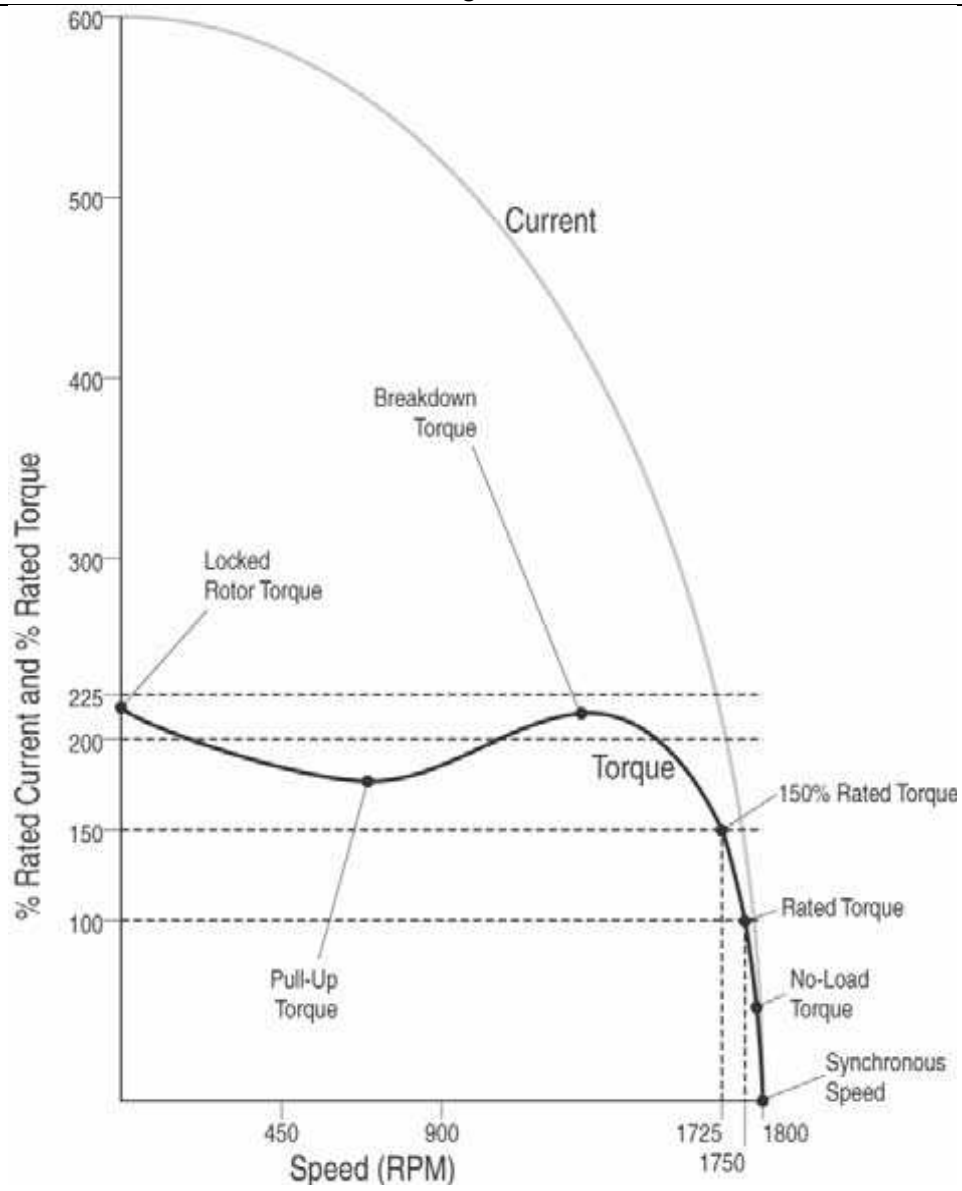


Figura 82 - Curvas de torque e de corrente de um motor trifásico genérico.

Fonte: (Rockweel Automation Drives Engineering Handbook).

5 Controladores Programáveis na hierarquia da automação

Uma das formas teóricas (Webb e Greshock,1992) de classificar o grau de automação nos processos produtivos é da pirâmide da automação industrial, que é composta de 5 níveis:

- Nível 1 – Controle manual. Por exemplo, uma botoeira para acionar o motor de uma ponte rolante; Sensores de temperatura, pressão, nível, umidade, opacidade, PH, movimento; medidores de vazão etc.

- Nível 2 – Controladores. CPs (Controlador Programável), Inversores de freqüência e demais *drivers* de acionamento.

- Nível 3 – Envolve conectividade do controle. Como exemplo, temos os sistemas supervisórios.

- Nível 4 – É definido pela exploração dos sistemas M.R.P (Material Requirement Planning), Just-in-time e o MRP II (Manufacturing Resource Planning).

- Nível 5 – Sistemas E.R.P (Enterprise Resource Planning), Gestão de Recursos Corporativos), SAP, etc.

A figura 83 ilustra o modelo da pirâmide da automação industrial.

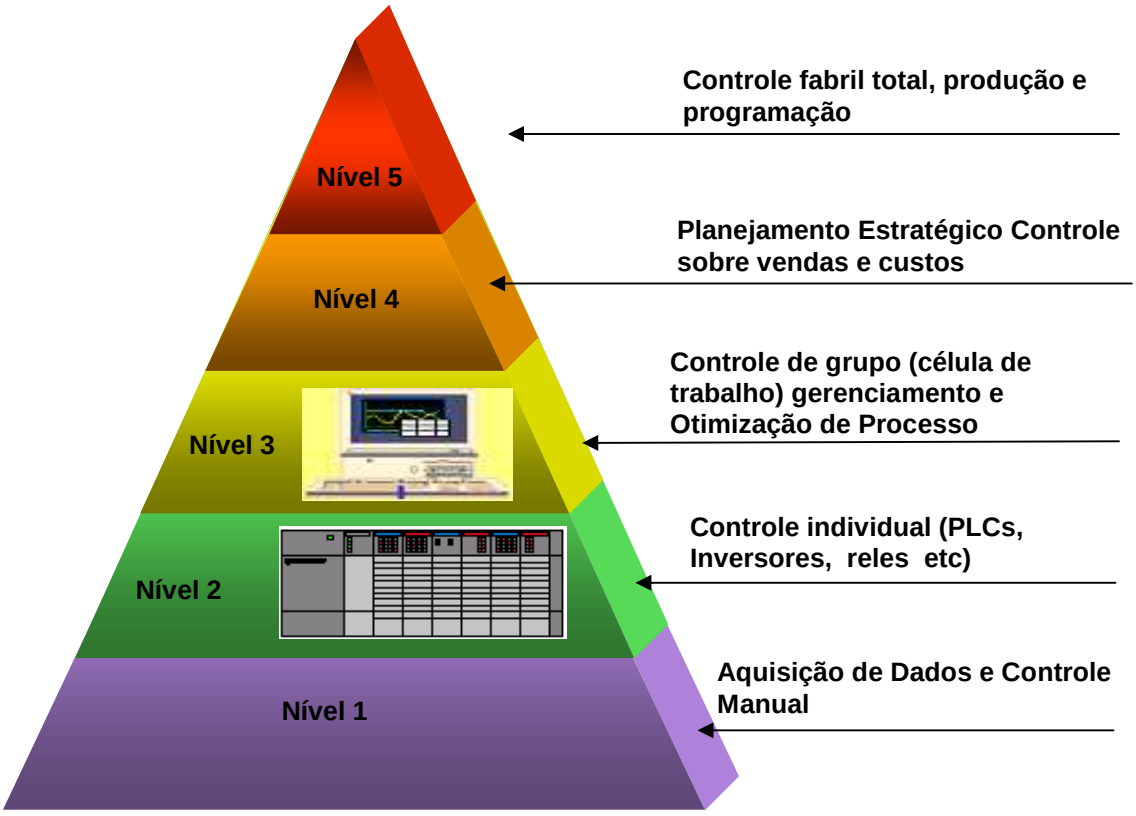


Figura 83 - Modelo da pirâmide da automação. Fonte (Andrade 2007).

Dentro desta classificação os CPs operam no nível 2 (estação). Os mesmos recebem sinais ou dados dos dispositivos de campo e atuam nos mesmos. Os CPs também recebem “set points” dos sistemas supervisórios, e enviam dados para os

sistemas mesmos. Os Sistemas Supervisórios (S.S.) operam no nível célula e recebem as informações de dispositivos controladores, tais como controladores lógicos programáveis (CLPs), inversores de freqüência, etc. Os sistemas supervisórios também operam como interface homem maquina (I.H.M.) com operadores humanos, para que os mesmos possam analisar, diagnosticar, tomar decisões e interferir no processo (Alves e Gebrael, 1998). As figuras 84 e 85 ilustram, respectivamente, uma arquitetura de hardware de um sistema genérico de automação, e a classificação de uma arquitetura de hardware genérica dentro do modelo da pirâmide da automação. A figura 86 ilustra a visão funcional da norma ISA 1995 sobre os diferentes níveis de atuação com dispositivos e sistemas de informação e automação.

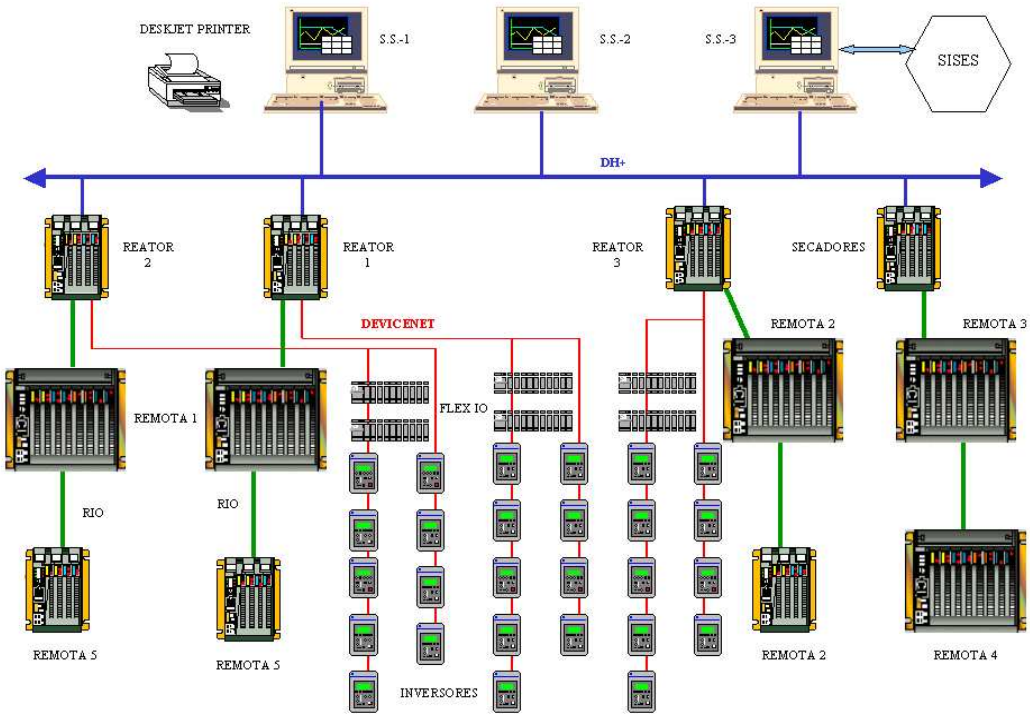


Figura 84 - Ilustrativo da arquitetura de hardware de um sistema genérico de automação.

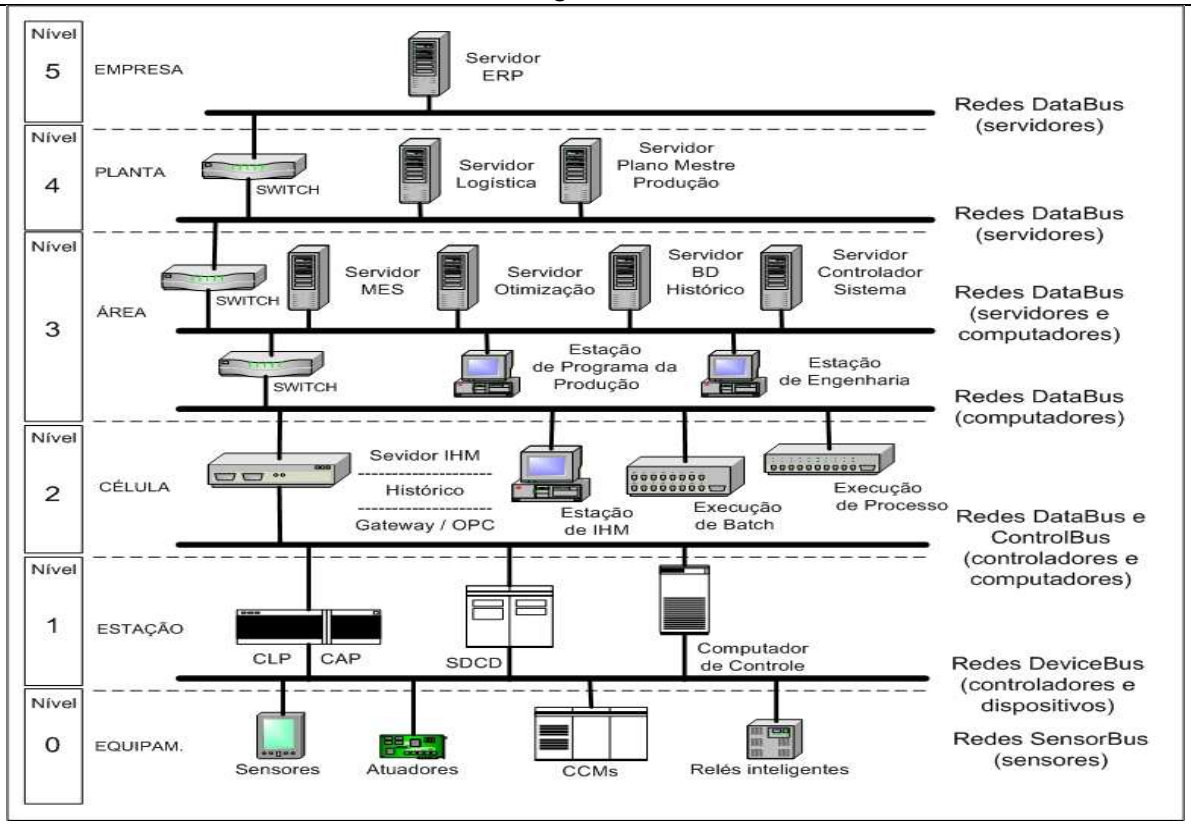


Figura 85 - Arquitetura de hardware genérica dentro do modelo da pirâmide da automação.

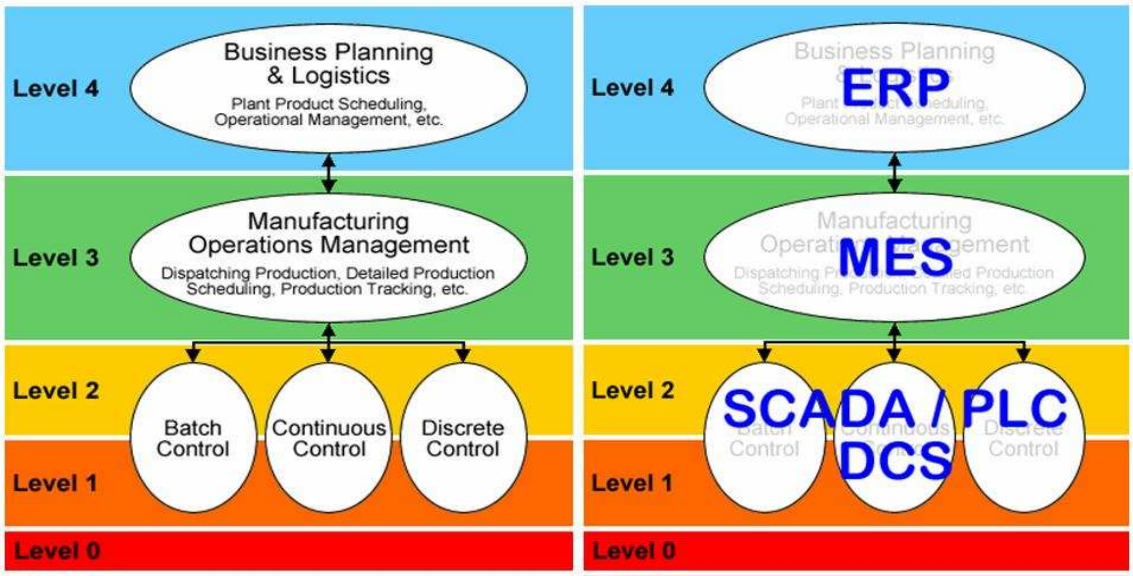


Figura 86 - Visão Funcional Fonte: (ISA 95).

6 Referências Bibliográficas

JOHN, K. H; TIEGELKAMP M.; IEC 61131-3 Programming Industrial Automation Systems – Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids Springer-Verlag Berlin Heidelberg 2010

STENERSON, J.; Programming ControLogix Programammble Automation Controllers Delmar,Cengage Learning 2009

LEWIS, R. W. Programming Industrial Control Systems Using IEC 1131-3. Institution of Eletrical Engineers, Londosn 1996.

MICHEL, G. Programmable Logic Controllers – Architecture and Applications. John Wiley & Sons Ltda. England 1990.

MORAIS C. C.; CASTRUCCI P. L. Engenharia de Automação Industrial - LTC Livros Técnicos e Científicos S.A. Rio de Janeiro 2001.

FRANCHI, C.M. Acionamentos elétricos Editora Érica Ltda 2008.

FRANCHI, C.M Inversores de frequência. Teoria e aplicações Editora Érica Ltda 2008.

MOHAN, N.; UNDELAND, T.; RBBINS,W.P.; Power Eletronics John Wiley &Sons, Inc 2003

PEREIRA, S. L.; ANDRADE, A. A. Módulo III – Automação I: Controladores Programáveis Apostila PECE 2005

MORAES, C.C.; SENGER, E.; Apostila Controladores Lógicos Programáveis EPUSP 1995)

PEREIRA, S. L.; Apostila Automação Industrial e Controladores Lógicos Programáveis EPUSP 2008

MATAKAS, L. PEREIRA, S. L.; Controladores Lógicos Programáveis Pontifícia Universidade X Católica São Paulo 2006

Manual Rockwell Automation: Obtendo Resultado com RS Linx TM, e Obtendo Resultado com RS Logix TM

Rockweel Automation Drives Engineering Handbook

Rockweel Automation Logix5000 Controllers, Catalog Numbers 1756 ControlLogix, 1756 GuardLogix, 1768 CompactLogix, 1768 Compact GuardLogix,1769, CompactLogix, 1789 SoftLogix, PowerFlex with DriveLogix Quick Start.

Rockweel Automation Logix5000 Controllers Sequential Function Charts Catalog Numbers 1756 ControlLogix, 1769 CompactLogix, 1789 SoftLogix, 1794 FlexLogix, PowerFlex 700S with DriveLogix Programming Manual

Rockweel Automation Logix5000 Controllers Function Block Diagram Catalog Numbers 1756 ControlLogix, 1769 CompactLogix, 1789 SoftLogix, 1794 FlexLogix, PowerFlex 700S with DriveLogix Programming Manual

Manuais on-line sobre a plataforma SLC 500 / MicroLogix disponibilizados no site mundial da Rockwell Automation - http://www.ab.com/catalogs/