UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

Elementos para Implementação de Sistemas Automatizados de Produção

62/99

Autor: **João Marcelo Georgini**
Orientador: **João Maurício Rosário**

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

Elementos para Implementação de Sistemas Automatizados de Produção

Autor: **João Marcelo Georgini**

Orientador: **João Maurício Rosário**

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

Dissertação de mestrado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 1999
S.P. - Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

G296e

Georgini, João Marcelo
Elementos para implementação de sistemas automatizados de produção / João Marcelo Georgini.-- Campinas, SP: [s.n.], 1999.

Orientador: João Maurício Rosário
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Automação. 2. Controle automático. 3. Sistemas homem-máquina. 4. Sistemas de tempo discreto. 5. Teoria das máquinas sequenciais. 6. Sistemas inteligentes de controle. I. Rosário, João Maurício. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

DISSERTAÇÃO DE MESTRADO

Elementos para Implementação de Sistemas Automatizados de Produção

Autor: **João Marcelo Georgini**

Orientador: **João Maurício Rosário**

Prof. Dr. João Maurício Rosário, Presidente
FEM/UNICAMP

Prof. Dr. Geraldo Nonato Telles
FEM/UNICAMP

Prof. Dr. José Reinaldo da Silva
USP/SÃO PAULO

Campinas, 16 de Dezembro de 1999

Dedicatória

Dedico este trabalho:

- à minha esposa Fátima, por seu amor, compreensão e, acima de tudo, cumplicidade em cada um de nossos projetos
- à nossa filha Marcella, nosso mais ousado 'projeto'
- aos meus pais, João Humberto (*in memorian*) e Olga, pela formação que me deram

Agradecimentos

Durante a execução deste trabalho, várias pessoas foram importantes. De modo especial, presto minha homenagem:

Ao meu orientador, prof. Dr. João Maurício Rosário, pela forma como conduziu este trabalho e principalmente pela confiança que depositou em mim.

Aos amigos Almiro Franco da Silveira Júnior, Eliseu Waterloo Storck, Luciano Fabricio, Marcelo Luís Bardini, Marcos Humberto Georgini, Marcos Vinicius Lopes e Wlamir Passos, pelo apoio e incentivo.

"Aquele que tenta perscrutar com humildade e perseverança os segredos das coisas,
ainda que disto não tome consciência, é como que conduzido pela mão de Deus,
que sustenta todas as coisas, fazendo que elas sejam o que são"
Gaudium et Spes 36

Resumo

GEORGINI, João Marcelo, *Elementos para Implementação de Sistemas Automatizados de Produção*, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1999. 223p. Dissertação (Mestrado)

Este trabalho aborda os aspectos referentes à descrição e implementação de sistemas automatizados de produção. São apresentados norma para descrição de sistemas seqüenciais (SFC - *Sequential Function Chart*) e equipamentos industriais para controle (PLC - Controlador Lógico Programável e Sistema Supervisório), visando a implementação de sistemas automatizados através de um modelo consistente que garanta conectividade e flexibilidade. Os conceitos apresentados são validados através de um exemplo de aplicação industrial, utilizando a Plataforma Industrial para Pesquisa, Ensino e Formação em Automação (PIPEFA), representando um sistema automatizado de produção composto por postos operacionais controlados por PLC e Sistema Supervisório. A apresentação didática desses elementos proporcionou a elaboração de uma referência bilbiográfica para utilização acadêmica e industrial.

Palavras Chave

Automação Industrial, Sistemas Discretos, Sistemas Seqüenciais, SFC (*Sequential Function Chart*), Grafcet, PLC, CLP, Linguagem Ladder, Sistema Supervisório, Integração de Sistemas

Abstract

GEORGINI, João Marcelo, *Elements for Implementation of Automated Manufacturing Systems*, Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1999. 223p. Dissertação (Mestrado)

This work approaches the aspects relative to description and implementation of automated manufacturing systems. International Standard for description of sequential systems (*SFC – Sequential Function Chart*) and industrial equipment for controlling (*PLC – Programmable Logical Controller* and *SCADA – Supervisory Control And Data Acquisition*) are described aiming the implementation of automated systems based on consistent model, allowing connectivity and flexibility. Approached concepts are validated through an industrial application example, under Industrial Platform to the Research, Education and Formation in Automation (*PIPEFA – 'Plataforma Industrial para Pesquisa, Ensino e Formação em Automação*), that represents an automated manufacturing system with manufacturer cells controlled by PLC and SCADA. Didactic presentation of these elements originated a bibliographical reference for academic and industrial use.

Key Words

Industrial Automation, Discrete Systems, Sequential Systems, SFC, Grafcet, PLC, Ladder Language, SCADA, System Integration

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este capítulo apresenta uma breve consideração sobre a automação industrial nas empresas, justificando sua necessidade e apontando as dificuldades normalmente encontradas em sua implantação. Apresenta, também, os objetivos e o delineamento deste trabalho.

Nos últimos anos, a indústria em todo o mundo sofreu grandes modificações devido à globalização; esta nova realidade tem forçado a modernização dos parques industriais em muitos países, tornando-os mais competitivos. Desta forma, para que uma empresa possa competir atualmente no mercado internacional, e até mesmo no nacional, é necessário que seus produtos e serviços tenham qualidade e preços acessíveis ao consumidor.

A automação é o meio de se atingir este objetivo. Porém, um sistema industrial moderno, competitivo, visando produtividade e qualidade, precisa ser planejado, usando tecnologia apropriada, com integração das diversas partes, proporcionando-lhe flexibilidade.

No Brasil, o cenário não é diferente. Durante um longo período, os produtos nacionais foram protegidos, fazendo com que o consumidor não tivesse muitas opções de fornecedores para determinados tipos de produtos. Assim, sem a competitividade, as empresas nacionais não investiram suficientemente em tecnologia para modernização de suas fábricas, não se preocupando em oferecer produtos de melhor qualidade. Porém, com a abertura do mercado brasileiro aos produtos estrangeiros, o consumidor passou a ter uma grande variedade de

fornecedores para um mesmo tipo de produto. Dentro deste novo ambiente, o consumidor ficou mais exigente, buscando mais qualidade aliada ao menor custo.

A solução é adaptar-se às exigências do mercado, cada vez mais competitivo. Com isso, a indústria nacional vem investindo na modernização de suas fábricas. Ao mesmo tempo, novas fábricas totalmente automatizadas se instalam no país, destacando-se as indústrias do setor automotivo.

A complexidade crescente dos Sistemas Automatizados de Produção implica grande dificuldade na definição clara das especificações funcionais associadas a estes sistemas. Por ser uma área multidisciplinar, a automação industrial envolve uma grande variedade de assuntos de fundamental importância, incluindo o desenvolvimento de redes de comunicação, softwares dedicados, integração e flexibilidade de sistemas.

Isto é percebido no momento da aquisição de equipamentos sofisticados por pequenas e médias empresas que, com expectativa de retorno a curto prazo e sem a devida adequação do ambiente fabril, têm experiências negativas (frustrações), adiando e prejudicando o ingresso de sistemas automatizados na linha de produção. Porém, vale a pena ressaltar que o processo de automatização da manufatura ainda é um movimento em implantação e em evolução, tanto em nosso país quanto no exterior, que exige a formalização para os modelos de Sistemas Automatizados.

Dentro dessa realidade, foi elaborado o projeto internacional de cooperação científica CAPES-COFECUB "Métodos e Ferramentas em Automação Industrial e Produção para o Desenvolvimento da Qualidade e da Produtividade nas PME/PMI", descrito por Rosário (1996), envolvendo o LAR (Laboratório de Automação Integrada e Robótica) da UNICAMP, o IA (Instituto de Automação) do CTI (Centro Tecnológico para Informática) e o LIISI (Laboratorie d'Ingénierie Integrée des Systèmes Industriels) da França. Foi construída, tanto no LAR como LIISI, uma Plataforma Industrial de Automação, representando um Sistema Automatizado de Produção (SAP), na qual pudessem ser implementadas e validadas ferramentas industriais de automação.

Neste trabalho serão abordados os aspectos referentes à descrição e implementação de SAPs, através da apresentação de norma para descrição de sistemas seqüenciais (*SFC – Sequential Function Chart*), apresentação de equipamento industrial para controle (PLC) e sua programação, e apresentação de Sistemas Supervisórios, visando a integração de sistemas através de um modelo consistente que garanta conectividade e flexibilidade.

1.1 Objetivos

O objetivo principal deste trabalho é o estabelecimento de conceitos, metodologias e ferramentas para a integração de sistemas automatizados. Com esta finalidade, as seguintes etapas deverão ser atingidas:

1) Apresentar didaticamente ferramentas para descrição e implementação de SAPs, levando-se em consideração normas existentes e equipamentos industriais (hardware e software), proporcionando a elaboração de uma referência bilbiográfica para utilização acadêmica e industrial
2) Apresentar um exemplo de aplicação utilizando a integração dessas ferramentas em um sistema industrial

1.2 Delineamento do Trabalho

Esta dissertação de mestrado está dividida da seguinte forma:

- Capítulo 1 – Introdução: é feita uma consideração sobre a automação industrial nas empresas, justificando sua necessidade e apontando as dificuldades; e uma apresentação geral do trabalho, incluindo os objetivos e o delineamento do mesmo

- Capítulo 2 – Descrição de Sistemas Automatizados: é apresentada a norma internacional para preparação de Diagramas Funcionais Seqüenciais

- Capítulo 3 – Controlador Lógico Programável - *PLC*: é apresentada a arquitetura básica de um Controlador Lógico Programável, as possíveis configurações, linguagens de programação e o sistema de operação

- Capítulo 4 – Programação de PLCs – *Linguagem Ladder*: são apresentados conceitos de programação dos PLCs através da Linguagem *Ladder*, apresentando instruções básicas e programação por estágios

- Capítulo 5 – Sistemas Supervisórios em Automação Industrial – *Software Interact*: são apresentados conceitos e campos de aplicação dos sistemas supervisórios, com apresentação de um software comercial

- Capítulo 6 – Exemplo de Aplicação – *Projeto PIPEFA*: são descritos os elementos do projeto da Plataforma Industrial para Pesquisa, Ensino e Formação em Automação (PIPEFA), e apresentada a implementação realizada através da aplicação dos conceitos abordados

- Capítulo 7 – Conclusões e Perspectivas Futuras: são apresentadas as considerações finais e sugestões para trabalhos posteriores

Capítulo 2

Descrição de Sistemas Automatizados

Este capítulo apresenta os principais aspectos da Norma Internacional para "Preparação de Diagramas Funcionais Para Sistemas de Controle" – IEC 60848 (1988). São apresentados os elementos e as respectivas simbologias, além das regras necessárias à sua implementação. São utilizadas as seguintes referências bibliográficas: Chiacchio (1996), IEC (1998) e Roussel (1993).

De acordo com a definição do Vocabulário Eletrotécnico Internacional, capítulo 351: Controle Automático – IEC 50(351), um Sistema de Controle pode ser dividido em duas partes interdependentes:

- *Sistema Controlado* – sistema que executa a operação física
- *Equipamento de Controle* – equipamento que recebe informações provenientes do operador, do processo a ser controlado, etc, e emite ordens ao Sistema Controlado

Conforme o caso, a estrutura apresentada na figura 2.1 pode não ser totalmente aplicada a determinado sistema. Os itens apresentados em linha não-contínua (pontilhada) podem não ser necessários.

Por exemplo, o sistema de controle de uma prensa apresenta estrutura similar à descrita anteriormente, conforme figura 2.2. Já, o sistema de controle de um semáforo simples apresenta uma estrutura simplificada, conforme figura 2.3.

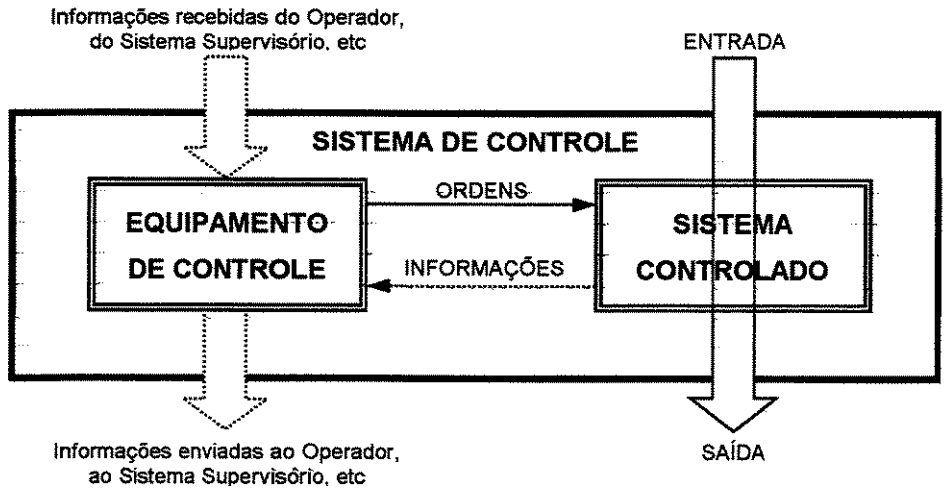


Figura 2.1 – Sistema de Controle

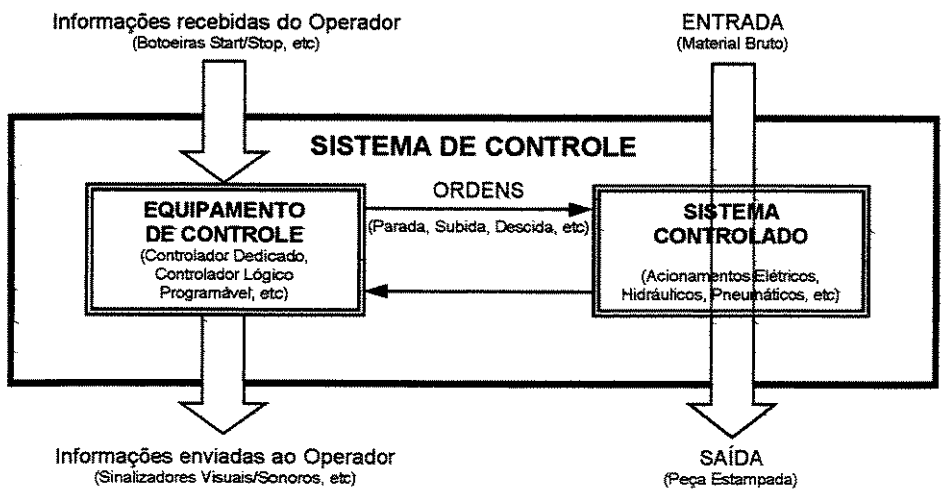


Figura 2.2 – Sistema de Controle de uma Prensa

A primeira fase do desenvolvimento de um Sistema Automatizado de Produção, independente de sua estrutura, consiste em descrevê-lo claramente. Esta descrição pode ser global (descrição do Sistema de Controle como um todo) ou específica (descrições individuais do Sistema Controlado e do Equipamento de Controle). Fornecer ao(s) projetista(s) as informações de forma detalhada, independente do nível da descrição, representa a maior dificuldade encontrada nesta fase.

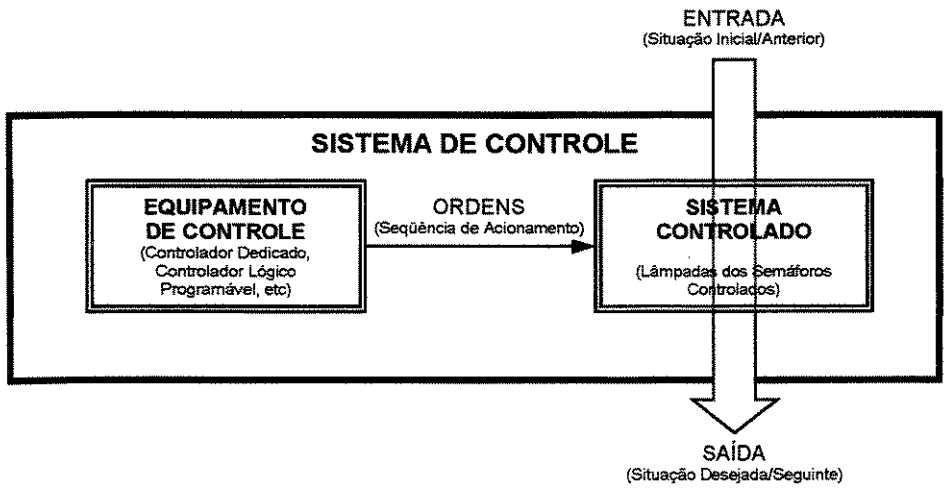


Figura 2.3 – Sistema de Controle de um Semáforo Simples

A linguagem verbal não é indicada, pois facilmente favorece interpretações diversas e até ambíguas. O texto descritivo tem utilização restrita, não sendo indicado a sistemas com ações complexas (decisões com várias possibilidades de progressões, ações simultâneas, etc).

Geralmente, as representações gráficas são de fácil compreensão. A dificuldade consiste em se encontrar uma representação aceita por todas as partes envolvidas no projeto, e que não seja diretamente relacionada à tecnologia ou implementação específicas. Além disso, algumas vezes pode ser difícil encontrar um símbolo gráfico para cada função a ser representada

O método de representação definido na Norma IEC 60848 combina símbolos gráficos e declarações textuais. Permite tanto uma descrição global do Sistema de Controle, como uma descrição específica do Sistema Controlado ou do Equipamento de Controle, através da relação precisa entre as entradas e saídas, independente das propriedades tecnológicas dos componentes usados ou de implementações futuras.

2.1 Origem da Norma – *"O Grafcet"*

Em 1975, um grupo de pesquisadores e gerentes industriais franceses, envolvidos em complexos sistemas de controle discreto, se reuniram para comparar e avaliar os modelos e métodos para construção de sistemas de controle seqüencial. Coletaram suas experiências

próprias, nas quais usavam dezessete técnicas diferentes: alguns utilizavam questionários empíricos, outros utilizavam modelos tecnológicos, outros ainda utilizavam modelos teóricos puros derivados de 'Máquinas de Estado' ou 'Redes de Petri'. Decidiram, então, construir um modelo customizado, mais fácil que os até então utilizados, e mais adequado aos sistemas complexos e particularmente aos sistemas de manufatura. Após dois anos de vários encontros de estudo e trabalho, propuseram um modelo chamado GRAFCET. O nome derivou-se de "*Graph*" – pois o modelo tinha um fundamento gráfico, e AFCET (*Association Française de Cybernétique Économique et Technique*) – associação científica que suportou todo o trabalho.

Os conceitos básicos deste sistema de controle discreto eram, e permanecem ainda hoje, extremamente claros e simples: a "etapa", a "ação", a "transição" e a "condição associada à transição". A etapa representa um estado parcial do sistema, no qual uma ação é realizada. Em determinado instante, uma etapa pode estar "ativa" ou "inativa". A ação associada somente é realizada se a etapa estiver ativa, e permanece inalterada se a etapa estiver inativa. A transição, que 'conecta' a etapa precedente (uma ou várias etapas) à etapa seguinte (uma ou várias etapas), representa uma decisão para mudança de estado do sistema (a ação da etapa precedente é seguida pela ação da etapa seguinte). Para que uma transição seja efetuada são necessárias duas condições:

- que a etapa (uma ou várias etapas) precedente à transição esteja ativa
- que a condição (boolena) associada à transição seja verdadeira

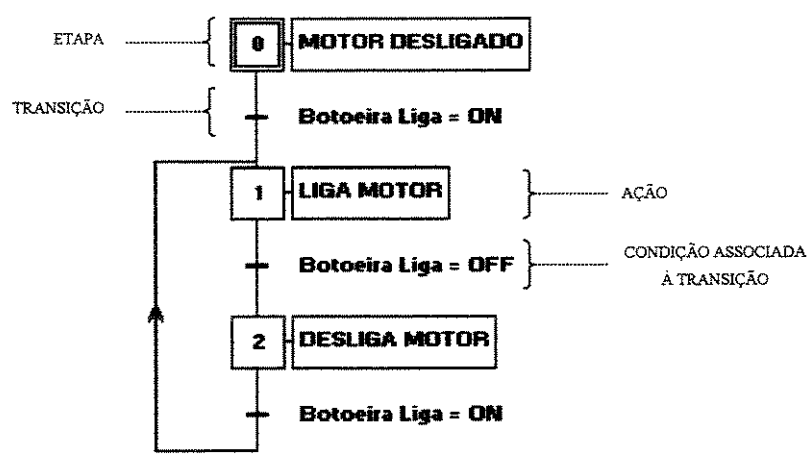


Figura 2.4 – Exemplo Simples de GRAFCET

Ao ser efetuada a transição, a etapa precedente (uma ou várias etapas) torna-se inativa e a etapa seguinte (uma ou várias etapas) torna-se ativa, conseqüentemente a ação associada à etapa precedente deixa de ser realizada e a ação da etapa seguinte passa a ser realizada. A figura 2.4 apresenta um exemplo de Grafcet.

Durante vários anos, este modelo foi testado em empresas privadas e instituições de ensino da França, revelando-se rapidamente ser muito eficiente à representação de pequenos e médios sistemas seqüenciais. Assim, foi introduzido nos programas educacionais franceses. Simultaneamente, foi proposto como uma norma à associação francesa AFNOR (*Association Française de Normalisation*), sendo aceito como tal em 1982. Então, os fabricantes de PLCs e os produtores de software escolheram o Grafcet como uma linguagem de representação para controles seqüenciais booleanos e propuseram implementações tanto para os PLCs como para os PCs. A utilização industrial aumentou. Pesquisadores começaram a estudar o uso teórico e prático deste modelo em métodos e ferramentas de projeto. Como conseqüência, surgiu a necessidade de normatização.

Em 1988, a IEC (*International Eletrotechnical Commission*) adotou o Grafcet como Norma Internacional sob o nome inglês "*Sequencial Function Chart*" – SFC (Diagrama Funcional Seqüencial), com o título "*Preparation of Function Charts for Control Systems*" (Preparação de Diagramas Funcionais Para Sistemas de Controle) e referência "IEC 848". Com a alteração nas referências das Normas IEC, ocorrida recentemente, passou a IEC 60848.

Embora tenha sido preparada visando aplicações eletrotécnicas, pode ser aplicada também a sistemas não-elétricos (hidráulicos, pneumáticos ou mecânicos, por exemplo), pois descreve as funções de controle relativas a determinado sistema, independente do campo de aplicação. O método de representação proposto serve como "ferramenta de comunicação" entre as diferentes áreas (disciplinas tecnológicas) envolvidas no desenvolvimento e utilização de Sistemas Automatizados. A seguir, são apresentados os elementos e as regras definidas na Norma IEC 60848.

2.2 Diagrama Funcional Seqüencial – *SFC* (*Sequential Function Chart*)

Os elementos do Diagrama Funcional Seqüencial são:

- **Etapas** – às quais são associadas as **Ações**
- **Transições** – às quais são associadas as **Condições**
- **Ligações Orientadas** – que conectam as etapas às transições, e estas às etapas

A combinação destes elementos proporciona uma representação 'estática' do Sistema Automatizado. Aplicadas as **Regras de Evolução**, obtem-se uma visão 'dinâmica' do mesmo.

2.2.1 Etapas

Cada etapa corresponde a uma condição invariável, e bem-definida, do sistema descrito. A eficiência e precisão de um Diagrama Funcional Seqüencial estão diretamente relacionadas à quantidade de etapas utilizadas para descrever determinado sistema. Portanto, quanto maior o número de etapas em que se puder dividí-lo, maior a eficiência da descrição de cada etapa e maior a precisão do Diagrama Funcional como um todo. A simbologia utilizada para representar uma etapa é apresentada na figura 2.5.

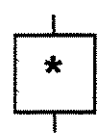


Figura 2.5 – Etapa (Simbologia Genérica)

Não é definida uma relação de tamanho entre os lados do retângulo, sendo recomendada a igualdade (quadrado). As etapas são referenciadas através de caracteres alfa-numéricos (em substituição ao asterisco, conforme apresentado na figura 2.6) de forma arbitrária, ou seja, não há necessidade de utilização seqüencial dos mesmos, e nem de respeito à ordem numérica (crescente ou decrescente). No entanto, não é permitido que etapas distintas tenham a mesma referência.

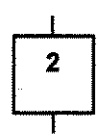


Figura 2.6 – Referência de uma Etapa

Etapa Ativa e Etapa Inativa

Em determinado instante, uma etapa pode estar ativa ou inativa, sendo que a situação de um sistema é determinada pelo conjunto de etapas ativas naquele momento.

Etapa Inicial

A etapa inicial é ativada incondicionalmente no início do controle de um sistema e indica a situação inicial do mesmo. Podem existir tantas etapas iniciais quantas se fizerem necessárias, sendo que todas serão ativadas simultaneamente no início do controle do sistema. A simbologia utilizada para representar uma etapa inicial é apresentada na figura 2.7.

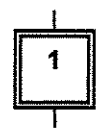


Figura 2.7 – Etapa Inicial

Indicação do Estado de uma Etapa

A representação do estado de uma etapa pode ser feita pelos valores lógicos "0" (inativa) ou "1" (ativa) de uma variável binária "X". Por exemplo X2 = 0 (etapa 2 inativa).

Para indicação das etapas ativas em determinado momento, no Diagrama Funcional, utiliza-se um ponto (•) localizado na parte inferior dos símbolos correspondentes, como apresentado na figura 2.8. Este ponto não pertence à simbologia da etapa, sendo utilizado apenas para análise e/ou apresentação do Diagrama Funcional Seqüencial.

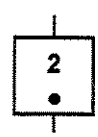


Figura 2.8 – Indicação de Etapa Ativa no Diagrama Funcional Seqüencial

2.2.2 Ações Associadas às Etapas

As ações são executadas somente se a etapa à qual estão associadas estiver ativa; caso contrário, são ignoradas. Enquanto a etapa estiver ativa, as ações podem ser iniciadas, continuadas ou finalizadas. Quando a etapa for desativada, as ações podem ser continuadas ou finalizadas, conforme a definição utilizada.

A ação associada à etapa é definida por declaração textual ou simbólica inserida em um retângulo, de qualquer tamanho (geralmente, utiliza-se a mesma altura do retângulo da etapa), conectado ao lado direito da etapa correspondente, conforme apresentado na figura 2.9.

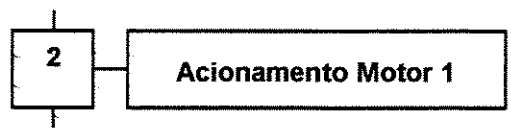


Figura 2.9 – Ação Associada à Etapa

Para representar mais de uma ação associada à mesma etapa, utiliza-se uma das formas apresentadas na figura 2.10. Embora as ações sejam definidas em retângulos individuais, a simbologia utilizada não especifica qualquer seqüência entre as ações associadas à etapa.

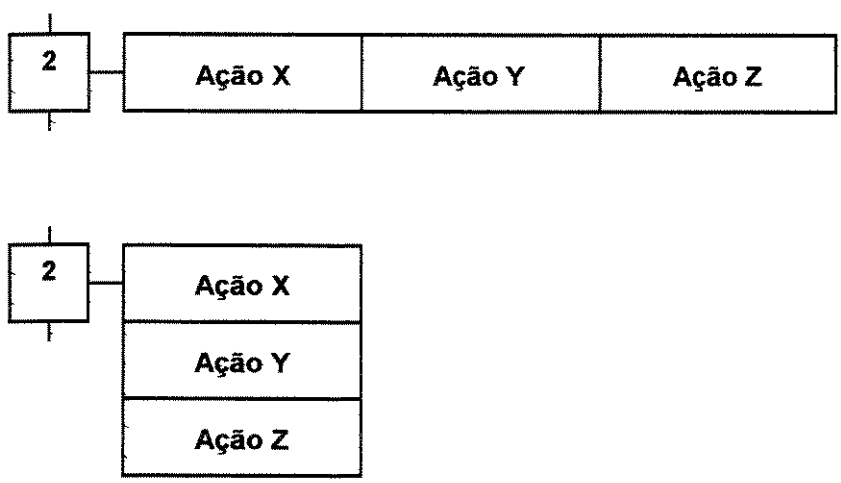


Figura 2.10 – Representação de Várias Ações Associadas a Mesma Etapa

Definição da Ação Associada à Etapa

É importante que a definição da ação associada à etapa seja feita com clareza e sem ambigüidades. Além de definir o comportamento do sistema em determinado momento, deve definir se a ação será mantida (continuada) ou finalizada após a desativação da etapa.

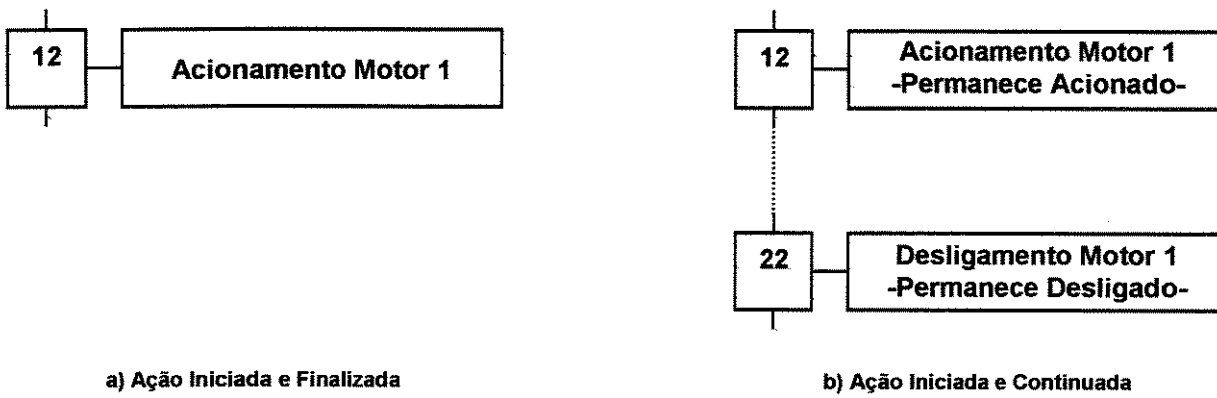


Figura 2.11 – Definição da Ação Associada à Etapa

A figura 2.11 apresenta dois exemplos de definição da ação associada à etapa:

a) a ação (Acionamento Motor 1) é iniciada ao ser ativada a etapa 12 e mantida enquanto esta etapa estiver ativa, sendo finalizada pela desativação da etapa 12

b) a ação (Acionamento Motor 1) é iniciada ao ser ativada a etapa 12 e continuada mesmo após a desativação desta (-Permanece Acionado-), sendo finalizada ao ser ativada a etapa 22 (Desligamento Motor 1 -Permanece Desligado-)

Ações Detalhadas (Qualificadas)

Como apresentado na figura 2.11a, uma ação não continuada é executada pelo período de tempo que a etapa estiver ativa. Porém, em alguns casos faz-se necessário condicionar ou limitar a execução da ação. Isto é possível através da utilização de ações detalhadas (ou qualificadas).

A simbologia utilizada para representar uma ação detalhada é apresentada na figura 2.12, na qual a simbologia de uma ação comum (figura 2.9) é sub-dividida em três campos distintos. O campo 'b' deve ter, no mínimo, o dobro da largura dos campos 'a' ou 'c'.

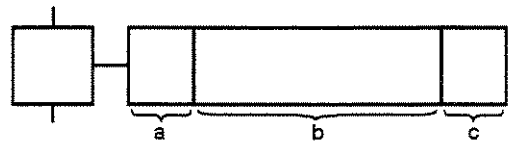


Figura 2.12 – Ação Detalhada Associada à Etapa

O campo 'a' deve conter o qualificador que define como a ação associada à etapa será executada. O campo 'b' deve conter a declaração textual ou simbólica da ação. E o campo 'c' deve conter a referência do sinal de retorno que será verificado pela transição seguinte. Os campos 'a' e 'c' somente são apresentados quando necessários.

São cinco os qualificadores definidos:

- **S** (*stored* – armazenada/mantida) – a ação é mantida (continuada) após a desativação da etapa (sem necessidade de especificação na declaração textual ou simbólica da ação) até ser finalizada por outra etapa, conforme apresentado na figura 2.13

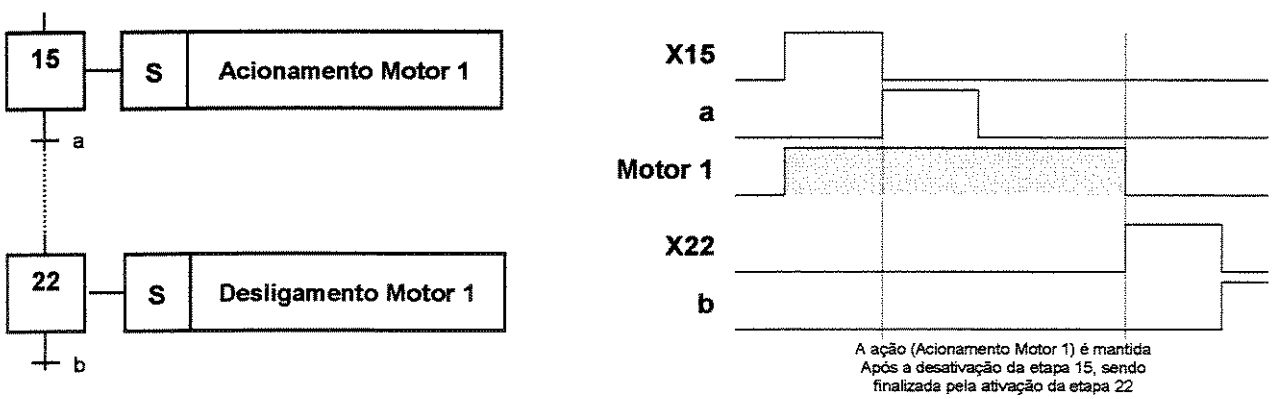


Figura 2.13 – Qualificador 'S' (*Stored*)

- **D** (*delayed* – atrasada) – a ação é iniciada após decorrido o tempo (atraso) especificado, e mantida enquanto a etapa estiver ativa, como apresentado na figura 2.14. Se a etapa permanecer ativada por um período menor que o especificado, a ação não é iniciada

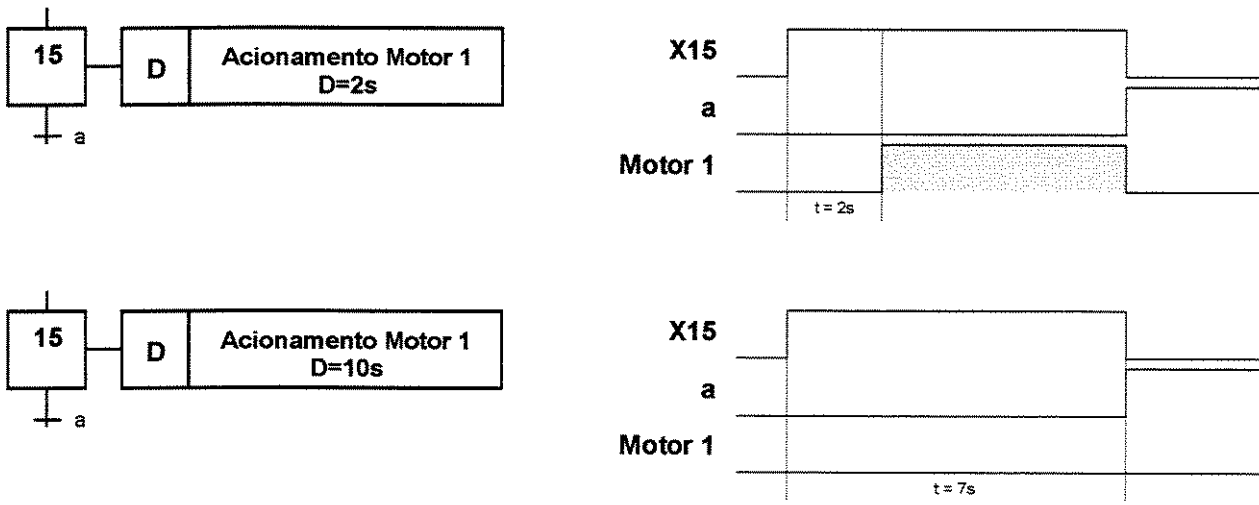


Figura 2.14 – Qualificador 'D' (*Delayed*)

- **L** (*time limited* – limitada pelo tempo) – a ação é iniciada e mantida enquanto a etapa estiver ativa ou até ser atingido o tempo especificado, como apresentado na figura 2.15

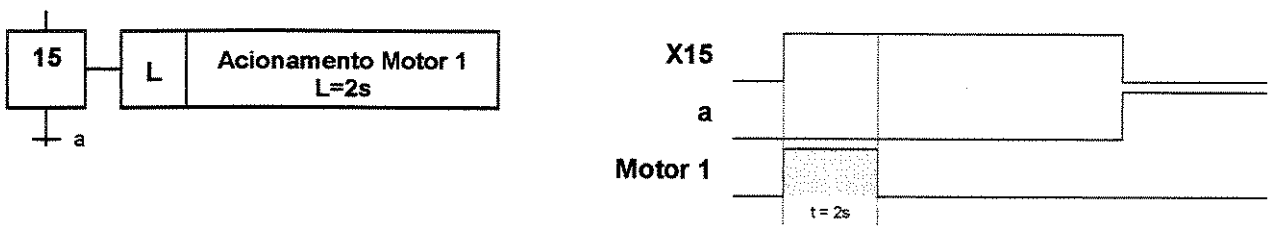


Figura 2.15 – Qualificador 'L' (*Limited Time*)

- **P** (*pulse shaped* – pulsada) – quando o tempo de execução da ação for muito pequeno, utiliza-se o especificador **P** em vez de **L**, como apresentado na figura 2.16

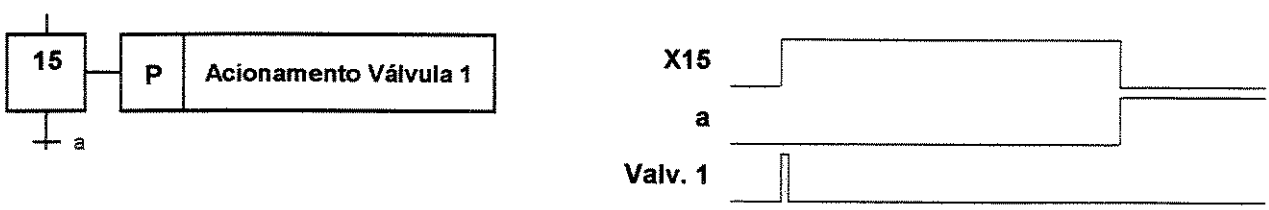


Figura 2.16 – Qualificador 'P' (*Pulse Shaped*)

- **C** (*conditional* – condicional) – a ação é iniciada e mantida enquanto a etapa estiver ativa, desde que a condição lógica especificada seja satisfeita (verdadeira), a qual pode ser indicada interna ou externamente ao símbolo, conforme apresentado na figura 2.17

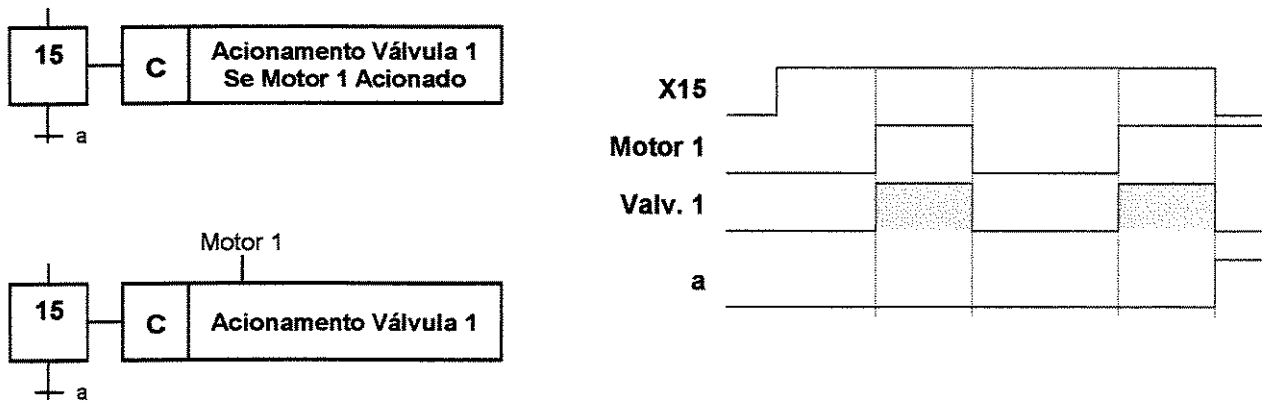
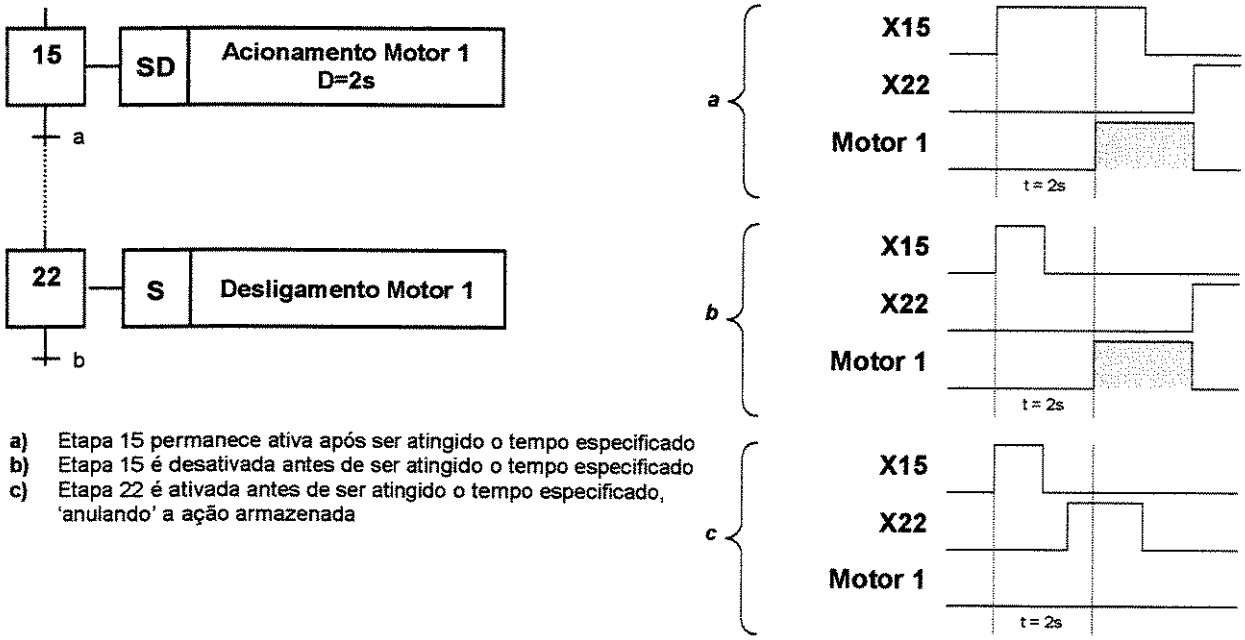


Figura 2.17 – Qualificador 'C' (*Conditional*)

Além de um único qualificador, uma ação pode ser detalhada através da combinação de qualificadores. Neste caso, a ordem de apresentação dos mesmos determina a seqüência que deve ser satisfeita para execução da ação associada à etapa. É possível utilizar qualquer combinação de qualificadores, excluindo-se 'LP' e 'PL' (funções de tempo). Alguns exemplos são apresentados a seguir.

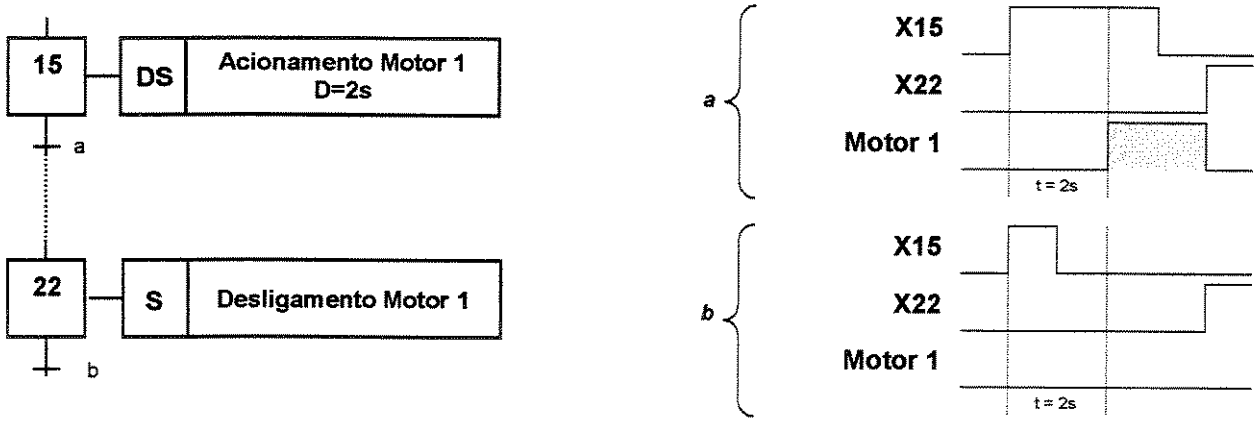
- **SD** (*Stored and delayed*) – a ação associada à etapa é armazenada ('S' – *stored*) e iniciada após o tempo especificado ('D' – *delayed*), mesmo que a etapa não esteja mais ativa. A ação é continuada até ser finalizada por uma etapa seguinte, conforme apresentado na figura 2.18

- **DS** (*delayed and stored*) – a ação associada à etapa é iniciada após o tempo especificado ('D' – *delayed*), e continuada ('S' – *stored*) até ser finalizada por uma etapa seguinte. Se a etapa permanecer ativada por um período menor que o especificado, a ação não é iniciada, conforme apresentado na figura 2.19



a) Etapa 15 permanece ativa após ser atingido o tempo especificado
b) Etapa 15 é desativada antes de ser atingido o tempo especificado
c) Etapa 22 é ativada antes de ser atingido o tempo especificado, 'anulando' a ação armazenada

Figura 2.18 – Qualificadores 'SD'



a) Etapa 15 permanece ativa após ser atingido o tempo especificado
b) Etapa 15 é desativada antes de ser atingido o tempo especificado

Figura 2.19 – Qualificadores 'DS'

- **CSL** (*conditional stored and time limited*) – a ação é iniciada desde que a condição lógica seja satisfeita ('C' – conditional) e mantida ('S' – *stored*) até ser atingido o tempo especificado ('L' – *time limited*), mesmo que a etapa não esteja mais ativa, ou até ser finalizada por uma etapa seguinte. Nota-se na figura 2.20, que a condição indicada

externamente ao símbolo é relacionada ao campo 'a' do mesmo, uma vez que refere-se aos qualificadores ('SL') e não diretamente à ação ('Acionamento Válvula 1')

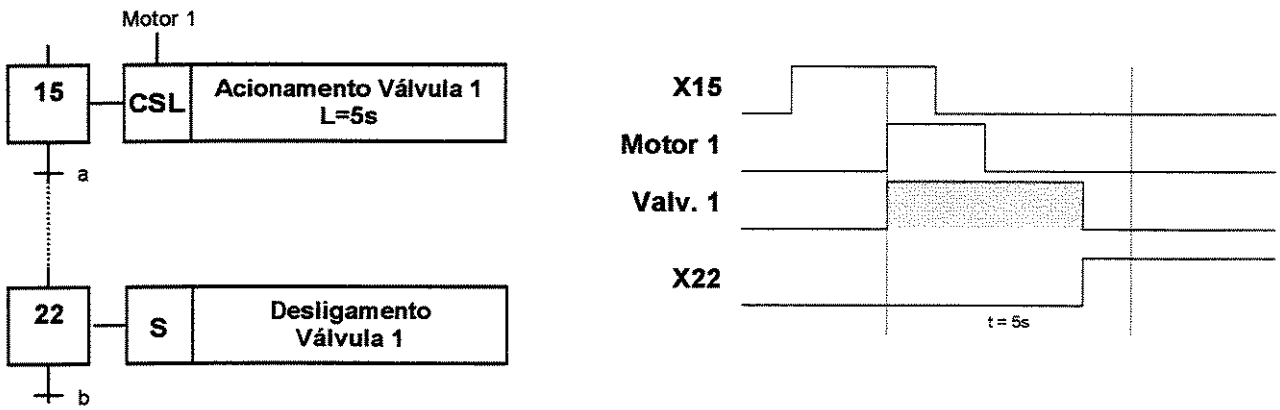


Figura 2.20 – Qualificadores 'CSL'

2.2.3 Transições

A possibilidade de evolução, ou transposição, de uma etapa ativa à outra é indicada por uma transição, a qual é representada por um pequeno traço entre a ligação das mesmas.

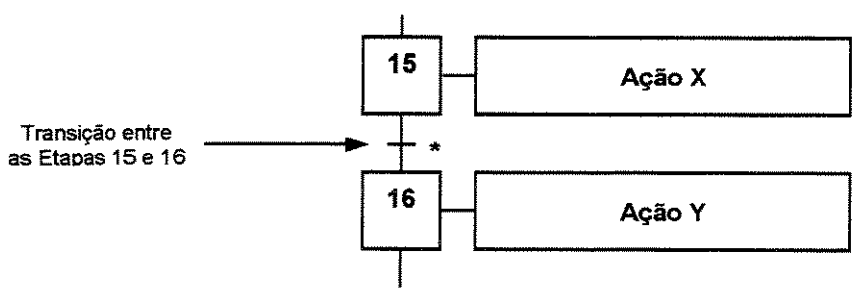


Figura 2.21 – Transição (Simbologia Genérica)

Às transições são associadas condições (em substituição ao asterisco apresentado na figura 2.21) que determinam a transposição de uma etapa à outra. Para que uma transição esteja habilitada (possível de ser transposta), é necessário que todas as etapas imediatamente precedentes (conectadas à transição em questão por meio de ligações orientadas) estejam ativas. E, para que a transição seja transposta, é necessário que esteja habilitada e a condição associada seja satisfeita (verdadeira).

A transposição de uma transição ocasiona a ativação de todas as etapas imediatamente seguintes e a desativação de todas as etapas imediatamente precedentes, simultaneamente. A figura 2.22 apresenta as condições necessárias para a transposição de uma transição, e o estado das etapas precedente e seguinte, antes e após a transição ser transposta.

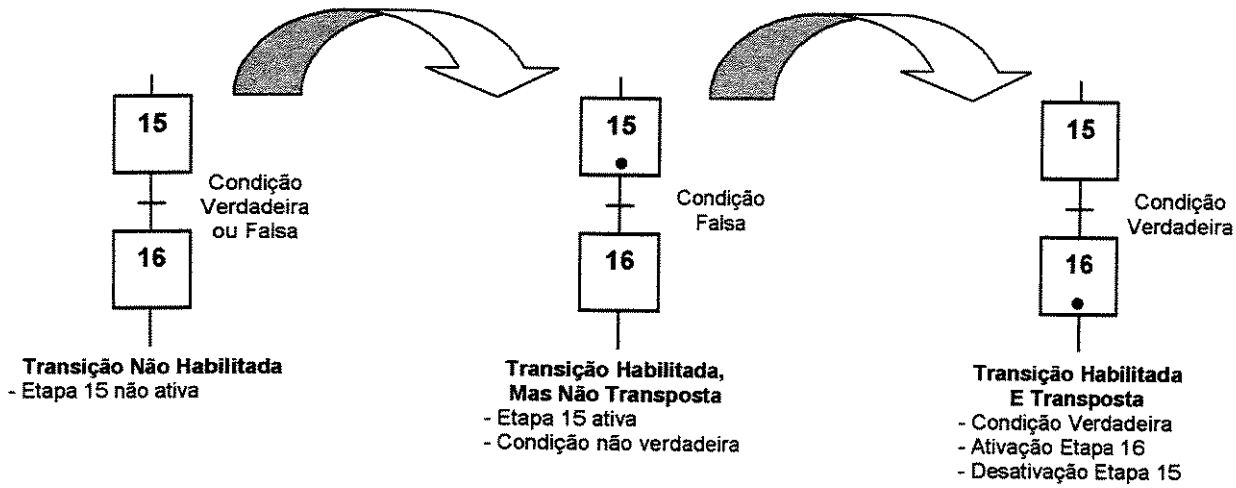


Figura 2.22 – Transposição de uma Transição

2.2.4 Condições Associadas às Transições

As condições associadas às transições são proposições lógicas, as quais podem ser verdadeiras ou falsas em determinado instante. Exprimem condições (internas ou externas) que devem ser satisfeitas para que a transição seja transposta. Podem ser representadas através de declarações textuais, expressões booleanas ou símbolos gráficos padronizados, colocados à direita ou à esquerda das transições, conforme apresentado na figura 2.23.

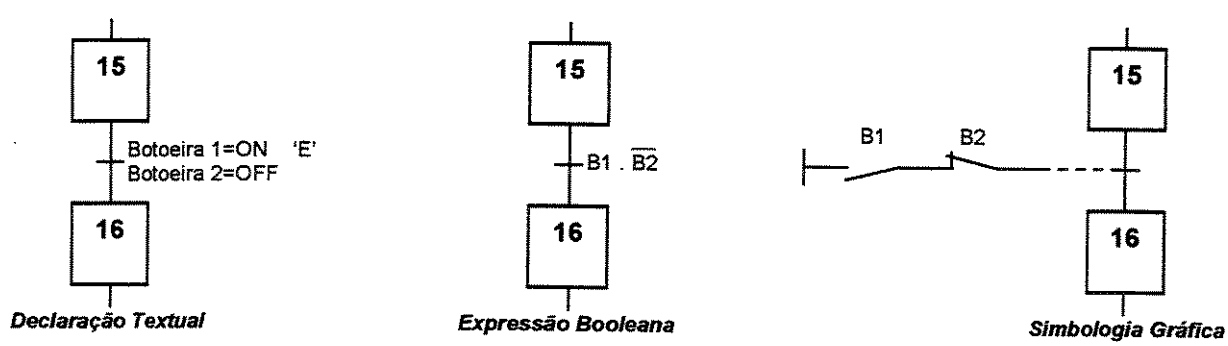


Figura 2.23 – Representação das Condições Associadas às Transições

Condições Detalhadas

As condições associadas às transições podem apresentar detalhes que as relacionem ao tempo ou ao estado lógico de uma variável.

Para representar uma condição relacionada ao tempo, através de declaração textual por exemplo, utiliza-se a notação "tempo/condição".

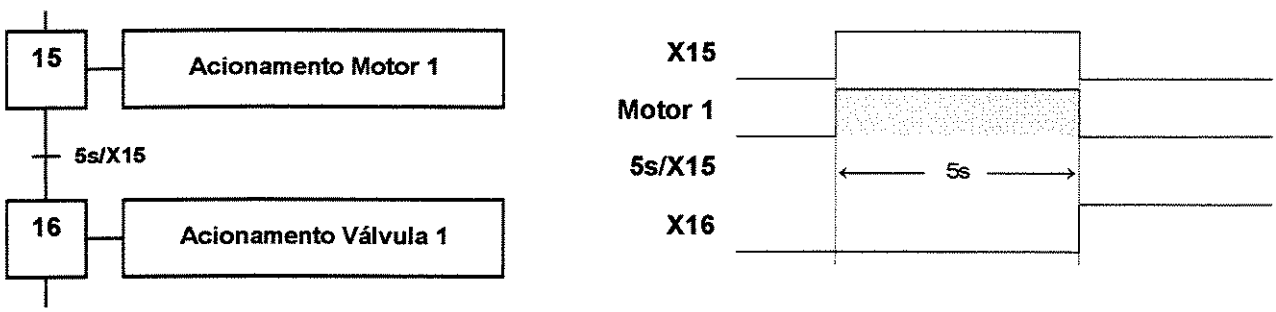


Figura 2.24 – Condição Relacionada ao Tempo

A condição apresentada na figura 2.24 indica que a transição deve ocorrer 5 segundos (tempo: '5s') após a Etapa 15 ter sido ativada (condição: 'X15').

Para representar uma condição relacionada ao estado lógico (0 ou 1) ou à transição (0→1 ou 1→0) de uma variável binária, recomenda-se a utilização das seguintes notações:

$\bar{a}$ ⇒ Estado Lógico 0 da Variável 'a'

a ⇒ Estado Lógico 1 da Variável 'a'

↑a ⇒ Transição 0→1 da Variável 'a'

↓a ⇒ Transição 1→0 da Variável 'a'

Transição Incondicional

À transição que deva ser transposta incondicionalmente, desde que habilitada, utiliza-se a notação "=1", conforme apresentado na figura 2.25.

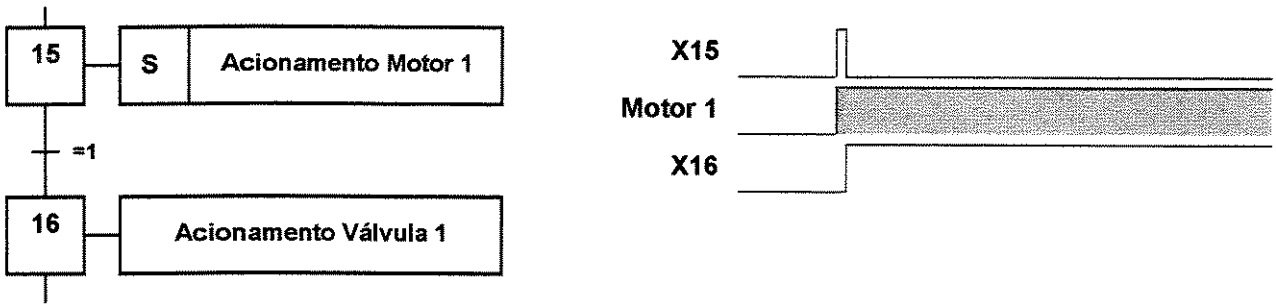


Figura 2.25 – Transição Incondicional

2.2.5 Ligações Orientadas

As etapas são conectadas às transições, e estas às etapas, através das ligações orientadas. De forma preferencial, são utilizadas vertical ou horizontalmente, podendo ser oblíquas para facilitar a compreensão do Diagrama Funcional.

Convencionalmente, o sentido de evolução é sempre de cima para baixo. Para indicar o sentido inverso (de baixo para cima), ou apenas para melhorar o entendimento em certas ocasiões, utilizam-se setas, conforme apresentado na figura 2.26.

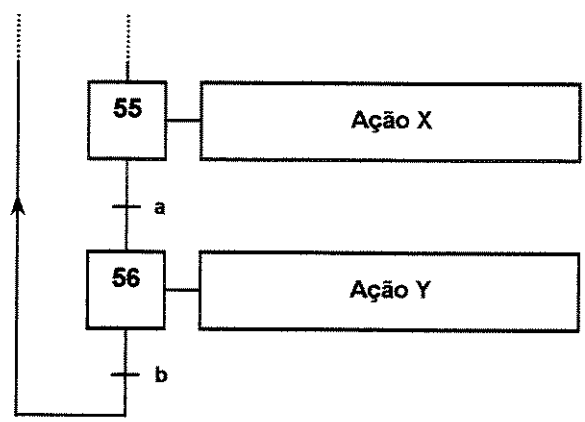


Figura 2.26 – Sentido das Ligações Orientadas

O cruzamento de ligações orientadas deve ser evitado. Quando necessário, utiliza-se preferencialmente uma das estruturas apresentadas na figura 2.27, nas quais é explícita a individualidade de cada ligação.

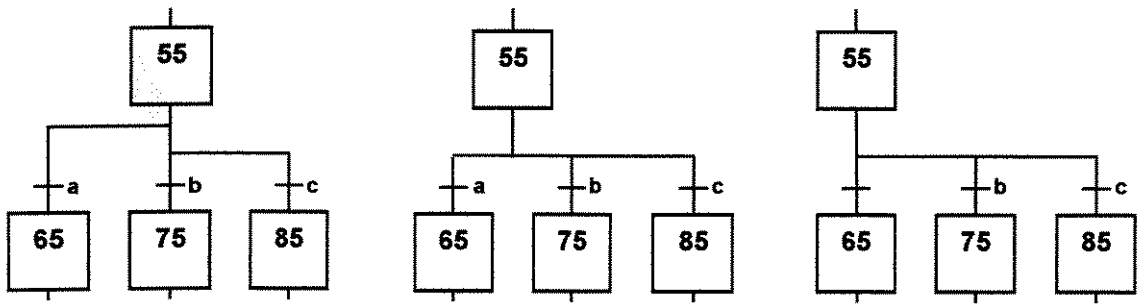


Figura 2.27 – Ligações Orientadas Não Cruzadas

Quando se necessita interromper uma ligação orientada para continuá-la em outra página, a referência da etapa seguinte e o número da página em que se encontra devem ser indicados, conforme apresentado na figura 2.28.

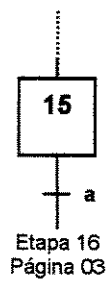


Figura 2.28 – Indicação de Continuidade da Ligação Orientada

Há duas regras simples para as conexões etapa-transição e transição-etapa através das ligações orientadas:

- duas etapas nunca podem ser conectadas diretamente. Portanto, devem ser separadas por uma única transição
- duas transições nunca podem ser conectadas diretamente. Portanto, devem ser separadas por uma única etapa

2.2.6 Regras de Evolução

A simples representação do Diagrama Funcional Seqüencial, através dos elementos apresentados, proporciona uma visão estática do Sistema Automatizado que se pretende

descrever. Aplicando-se as regras de evolução, apresentadas a seguir, obtém-se uma visão dinâmica do mesmo.

Regra 1: Situação Inicial

A situação inicial do sistema descrito é dada pelas etapas iniciais, que são ativadas incondicionalmente no início da operação do mesmo. Podem existir tantas etapas iniciais quantas forem necessárias. Obrigatoriamente, deve existir pelo menos uma etapa inicial em cada Diagrama Funcional Seqüencial.

Regra 2: Transposição de uma Tansição

Uma transição somente é transposta se:

- a mesma estiver habilitada (todas as etapas imediatamente precedentes ativadas)
- a condição associada for verdadeira

Regra 3: Evolução das Etapas Ativas

A transposição de uma transição ocasiona a ativação da(s) etapa(s) imediatamente seguinte(s) e a desativação da(s) etapa(s) imediatamente precedente(s), simultaneamente.

Regra 4: Transposição Simultânea de Transições

A representação de transições que serão transpostas simultaneamente deve ser feita através de linhas duplas agrupando as etapas imediatamente precedentes e as imediatamente seguintes, conforme apresentado na figura 2.29.

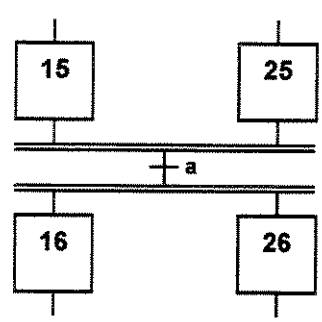


Figura 2.29 – Transições Simultâneas (I)

Caso tais transições estejam dispostas separadamente no Diagrama Funcional, utiliza-se a representação da figura 2.30. Em todas as transições simultâneas deve constar o asterisco (à esquerda da transição) e a referência da(s) etapa(s) envolvida(s).

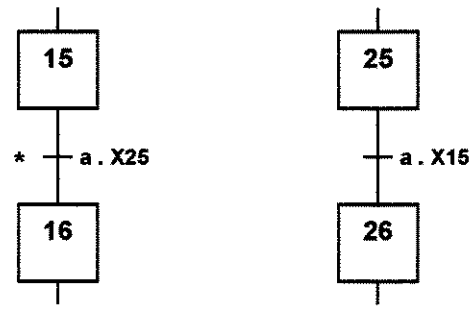


Figura 2.30 – Transições Simultâneas (II)

Regra 5: Condições Verdadeiras e Imediatamente Seguintes

Se, no instante de ativação de uma etapa, a condição de transição desta à etapa seguinte for verdadeira, a mesma não ocorrerá.

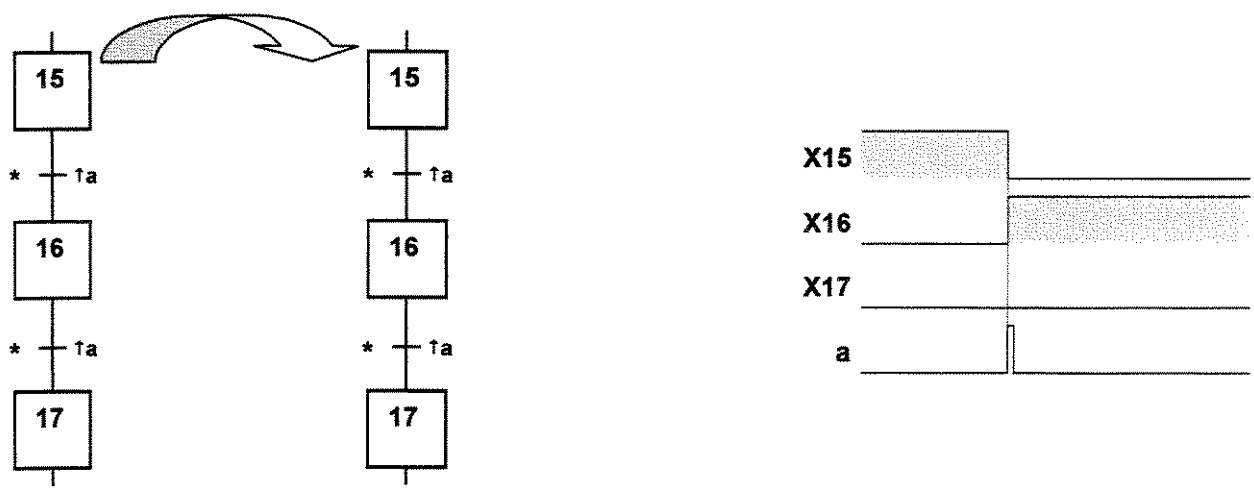


Figura 2.31 – Condições Verdadeiras e Imediatamente Seguintes

Conforme apresentado na figura 2.31, no instante que a condição de transição entre as etapas 15-16 e 16-17 se torna verdadeira (borda de subida da variável 'a'), a primeira transição apresentada está habilitada (etapa 15 ativa) e a segunda não (etapa 16 inativa). Conforme a 'Regra 2', apenas a transição entre as etapas 15-16 é transposta.

Regra 6: Ativação e Desativação Simultâneas de uma Etapa

Se ocorrer a ativação e desativação de uma mesma etapa simultaneamente, a ativação é prioritária.

Regra 7: Tempo Nulo

Os tempos para transposição de uma transição, ou ativação de uma etapa, podem ser extremamente curtos, geralmente dependentes da tecnologia utilizada para implementação do Sistema Automatizado. Porém, nunca podem ser considerados nulos, ou iguais a zero.

2.2.7 Estruturas Básicas

Além de seqüência simples, conforme apresentado na figura 2.32, é possível representar estruturas com seqüências seletivas (ou alternativas), seqüências simultâneas (ou paralelas), reutilização de uma seqüência, e ainda representar estruturas com detalhamento de uma etapa (sub-etapas).

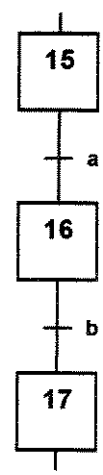


Figura 2.32 – Seqüência Simples

Seqüências Seletivas (Alternativas)

As seqüências seletivas são utilizadas para representar decisões, nas quais é definida apenas uma seqüência a ser seguida. As condições associadas às transições do início de cada seqüência devem ser exclusivas, ou seja, apenas uma condição pode ser verdadeira em determinado

instante. O início de uma seqüência seletiva é sempre divergente, e o final sempre convergente, conforme apresentado na figura 2.33.

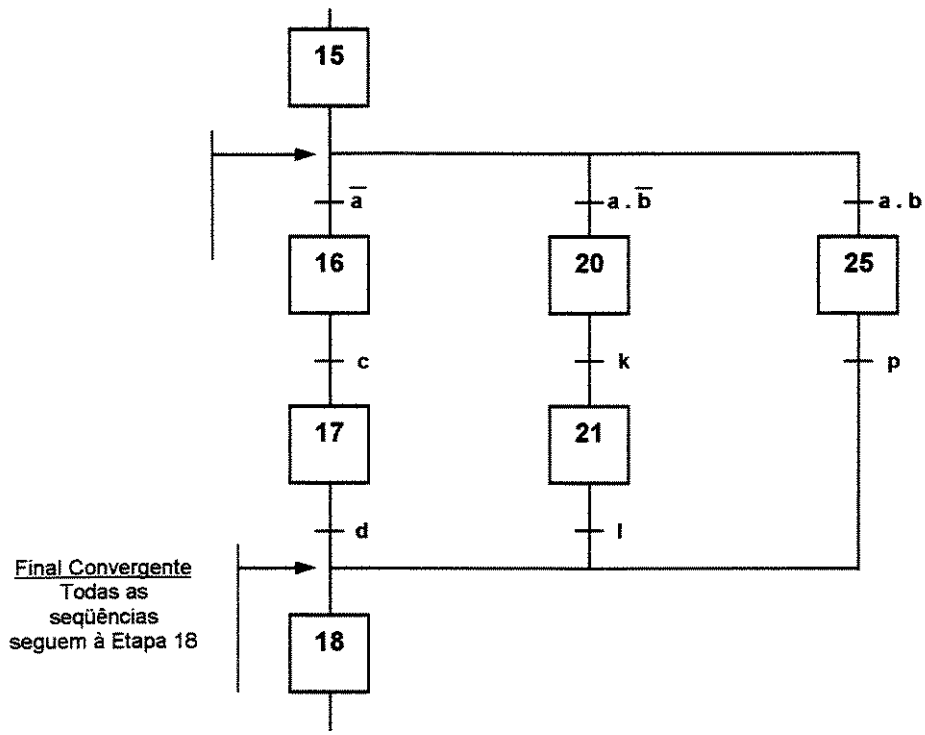


Figura 2.33 – Seqüência Seletiva (Alternativa)

Seqüências Simultâneas (Paralelas)

As seqüências simultâneas definem seqüências que são iniciadas paralelamente, através de uma única transição. Após a ativação, cada seqüência tem sua evolução independente. O início de uma seqüência simultânea é sempre divergente, e o final sempre convergente, ambos indicados por linhas duplas, conforme apresentado na figura 2.34.

Reutilização de uma Seqüência

Uma seqüência de etapas (simples, seletivas e/ou simultâneas) que apareça mais de uma vez no Diagrama Funcional e em situações distintas, pode ter a representação reutilizada, conforme apresentado na figura 2.35.

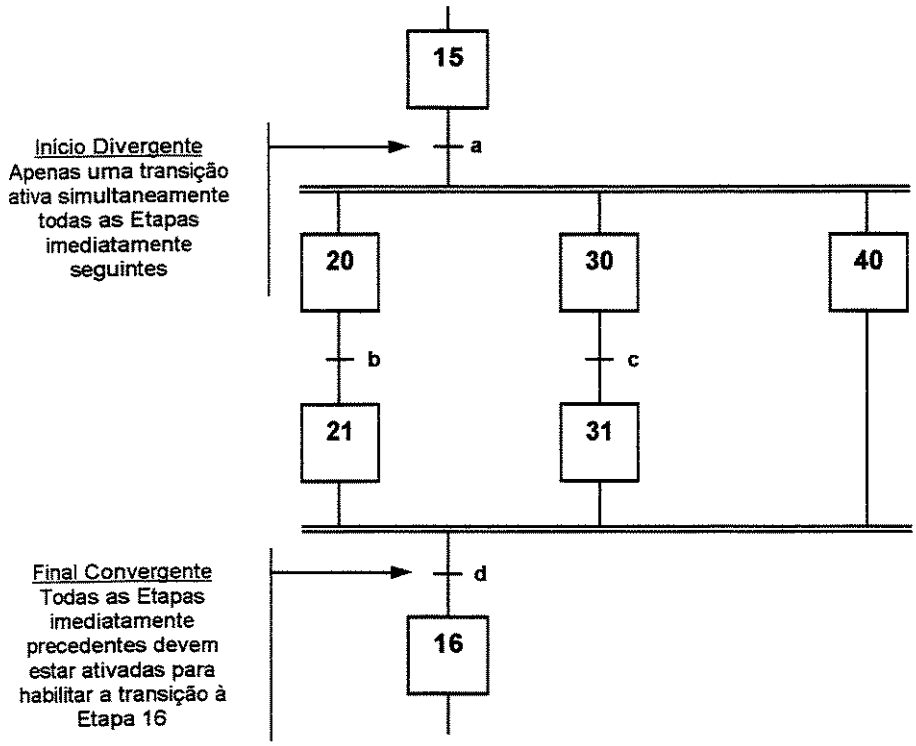


Figura 2.34 – Seqüência Simultânea (Paralela)

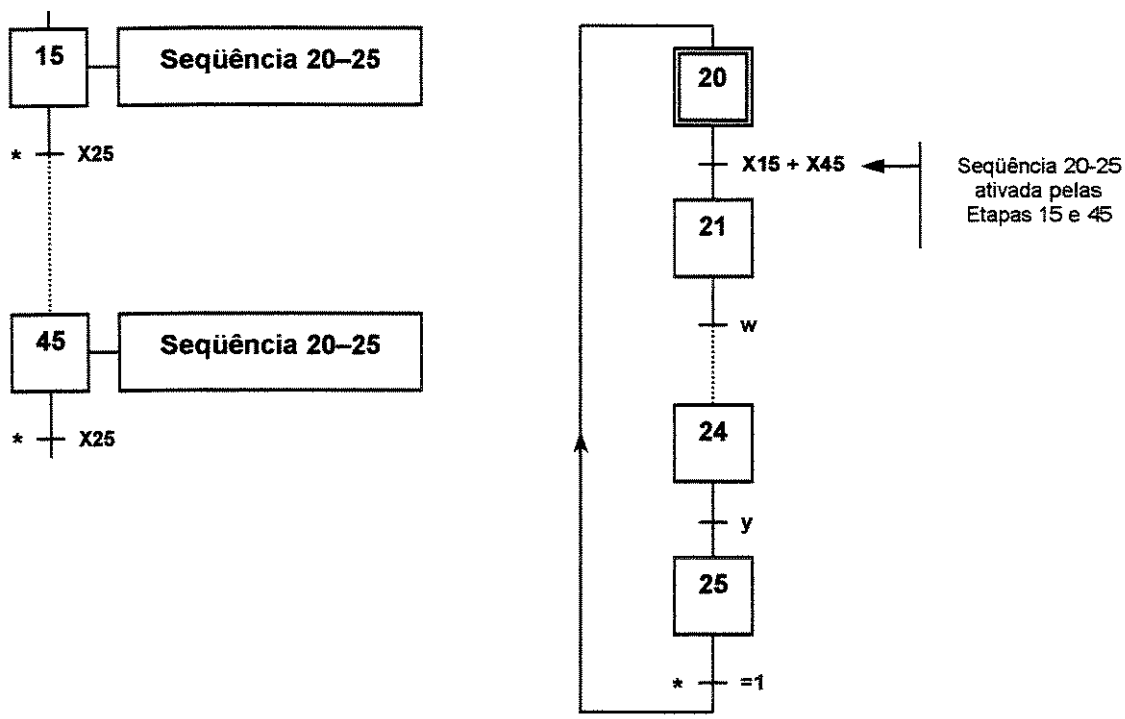


Figura 2.35 – Reutilização de uma Seqüência

Detalhamento de uma Etapa

Uma etapa pode ser apresentada com mais detalhes através de sub-etapas, as quais podem conter seqüencias simples, seletivas e/ou simultâneas, conforme apresentado na figura 2.36.

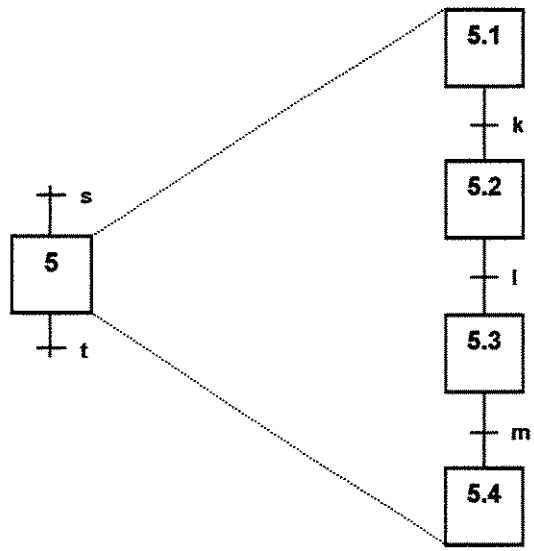


Figura 2.36 – Delhamento de uma Etapa

2.3 Exemplo de Aplicação

A figura 2.37 representa uma prensa hidráulica acionada bi-manualmente (segurança). Possui sensores para verificação da posição do punção (prensa alto e prensa baixo) e de presença de material.

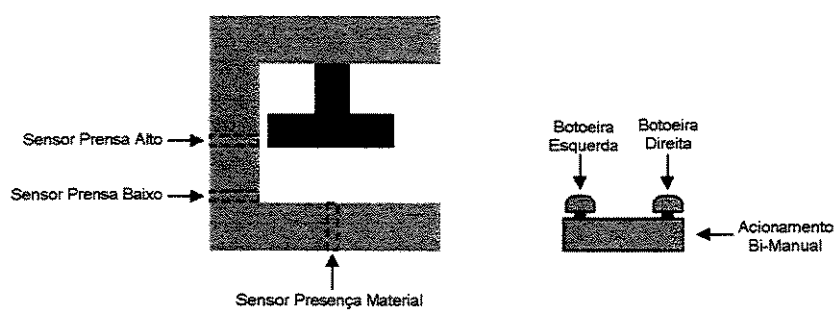


Figura 2.37 – Prensa Hidráulica

A figura 2.38 apresenta uma possibilidade de Diagrama Funcional Seqüencial relativo ao Sistema de Controle para este Sistema Automatizado.

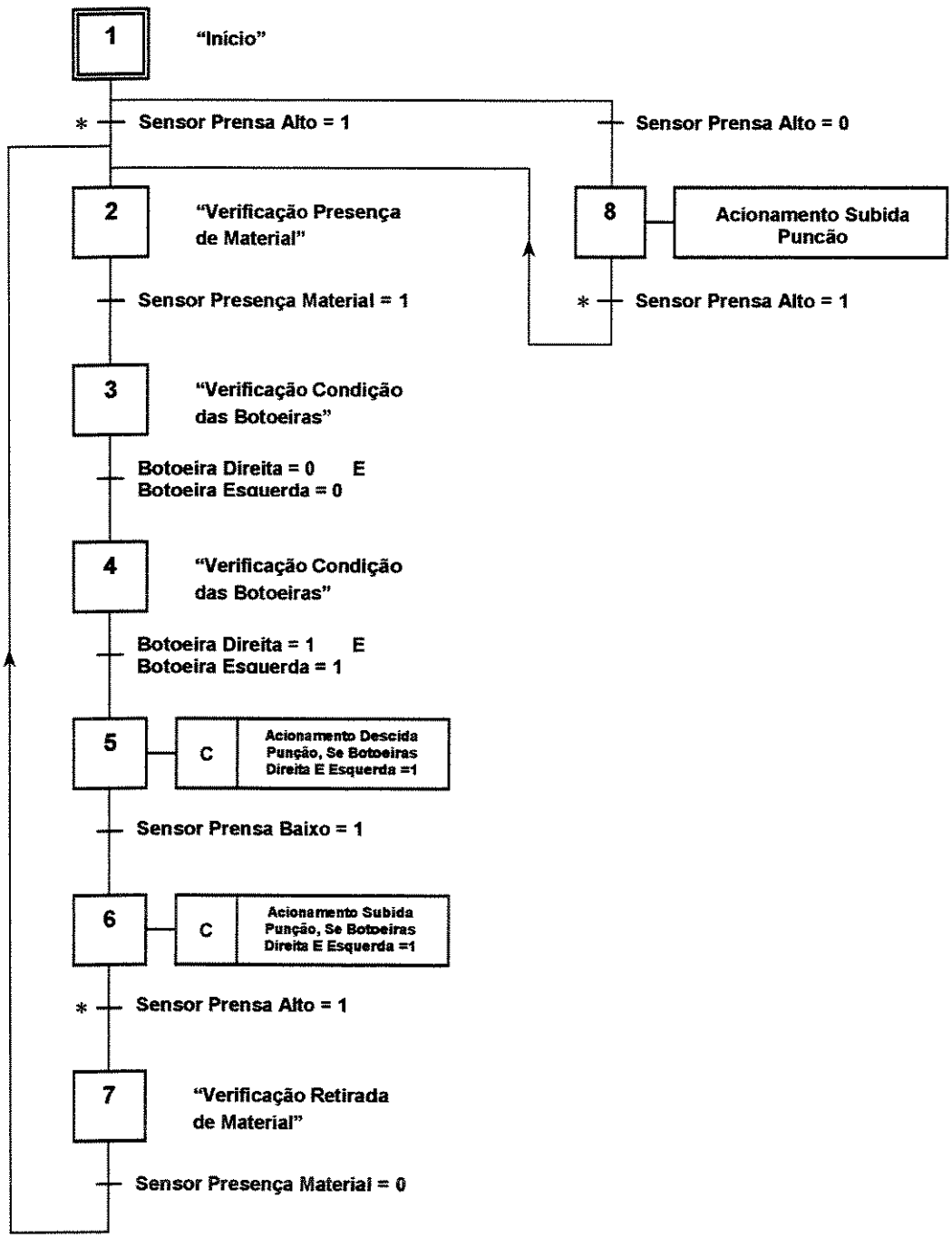


Figura 2.38 – Exemplo de Diagrama Funcional Seqüencial para Prensa Hidráulica

Neste capítulo foram apresentados os principais aspectos da Norma IEC 60848 (1988): seus elementos e respectivas simbologias, além das regras necessárias à sua implementação, sendo finalizado com a apresentação de um exemplo simples de aplicação.

No próximo capítulo será apresentada a arquitetura básica de um Controlador Lógico Programável, as principais linguagens de programação e o sistema de operação do mesmo.

Capítulo 3

Controlador Lógico Programável – *PLC*

Este capítulo apresenta a arquitetura básica de um Controlador Lógico Programável (PLC), descrevendo cada um de seus elementos. São apresentadas as possíveis configurações utilizadas em implementações de Sistemas Automatizados, as ferramentas e as linguagens de programação, além da descrição do sistema de operação interno. São utilizadas as seguintes referências bibliográficas: Bryan (1988), Celati (1995), Chiacchio (1996), PLC Direct (1997 e 1998) e Stenerson (1999). As fotos utilizadas são de produtos da *PLC Direct by Koyo*™.

O Controlador Lógico Programável, ou simplesmente PLC[1] (*Programmable Logic Controller*), pode ser definido como um dispositivo de estado sólido - um Computador Industrial, capaz de armazenar instruções para implementação de funções de controle (seqüência lógica, temporização e contagem, por exemplo), além de realizar operações lógicas e aritméticas, manipulação de dados e comunicação em rede, sendo utilizado no controle de Sistemas Automatizados de Produção. A figura 3.1 apresenta uma aplicação genérica com PLC.

Os PLCs surgiram para substituir os sistemas a relés, eliminando toda fiação para intertravamento dos relés de controle e facilitando as alterações necessárias.

---

[1] É adotado o termo inglês "PLC", por opção do autor. Porém, no mercado brasileiro normalmente são encontrados termos como "CLP" (Controlador Lógico Programável) e "CP" (Controlador Programável).

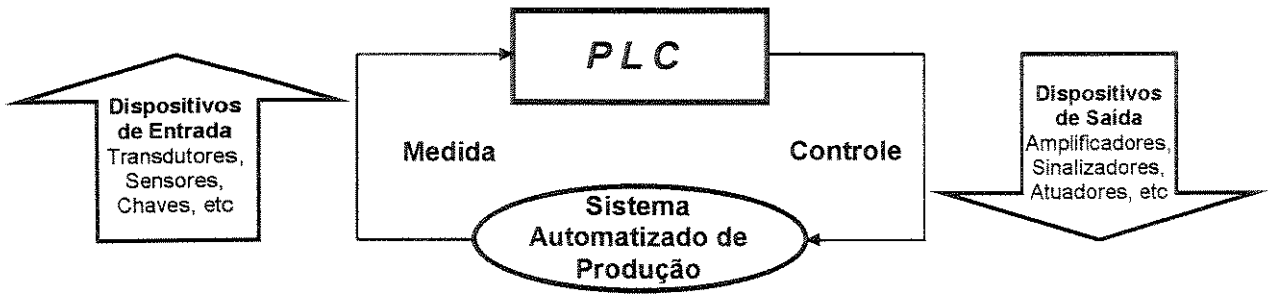
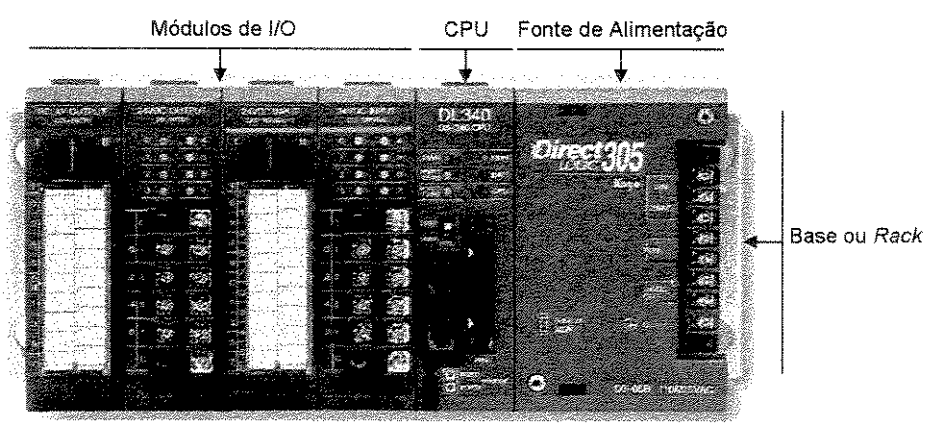


Figura 3.1 – Aplicação Genérica com Controlador Lógico Programável

Os principais blocos que compõem um PLC, conforme apresentado na figura 3.2, são:

- **CPU** (*Central Processing Unit* - Unidade Central de Processamento): compreende o processador (microprocessador, microcontrolador ou processador dedicado), o sistema de memória (ROM e RAM) e os circuitos de controle
- **Circuitos/Módulos de I/O** (*Input/Output* – Entrada/Saída): podem ser discretos (sinais digitais: 12VDC, 110VAC, contatos normalmente abertos, contatos normalmente fechados) ou analógicos (sinais analógicos: 4-20mA, 0-10VDC, termopar)
- **Fonte de Alimentação**: responsável pela tensão de alimentação fornecida à CPU e aos Circuitos/Módulos de I/O
- **Base** ou ***Rack***: proporciona conexão mecânica e elétrica entre a CPU, os Módulos de I/O e a Fonte de Alimentação. Contém o barramento de comunicação entre os mesmos, onde os sinais de dados, endereço, controle, e tensão de alimentação, estão presentes



PLC da Família DL305 – *PLC Direct by Koyo*

Figura 3.2 – Exemplo de PLC Disponível no Mercado

Pode ainda ser composto por **Circuitos/Módulos Especiais**: contador rápido (5kHz, 10kHz, 100kHz, ou mais), interrupção por hardware, controlador de temperatura, controlador de posicionamento de eixos, co-processadores (transmissão via rádio, controle Proporcional-Integral-Derivativo - PID, sintetizador de voz), e comunicação em rede, por exemplo.

A CPU executa a leitura dos *status* (condições, estados) dos dispositivos de entrada através dos Circuitos/Módulos de I/O. Esses *status* são armazenados na memória (RAM) para serem processados pelo Programa de Aplicação (desenvolvido pelo usuário e armazenado em memória RAM, EPROM ou EEPROM no PLC). Após a execução do Programa de Aplicação, o processador atualiza os *status* dos dispositivos de saída através dos Circuitos/Módulos de I/O, realizando a lógica de controle.

A programação do PLC é feita através de uma Ferramenta de Programação, que pode ser um Programador Manual (Terminal de Programação, *Handheld Programmer*), ou um PC com Software de Programação específico (ambiente DOS® ou Windows®). A Linguagem Ladder (*RLL - Relay Ladder Logic*, Lógica de Contatos de Relé), muito popular entre os usuários dos antigos sistemas de controle a relés, é a mais utilizada. Esta linguagem é a representação lógica da seqüência elétrica de operação, como ilustrado nas figuras 3.3 e 3.4.

A lógica implementada pelo PLC é muito similar à convencional, sendo que os dispositivos de entrada (elementos B0 e B1) são conectados ao Circuito/Módulo de Entrada, e o dispositivo de saída (elemento L0) ao Circuito/Módulo de Saída. O Programa de Aplicação determina o acionamento da saída em função das entradas (**B0 . B1 = L0**). Qualquer alteração desejada nesta lógica, é realizada através de alterações no programa, permanecendo as mesmas ligações nos Circuitos/Módulos de I/O.

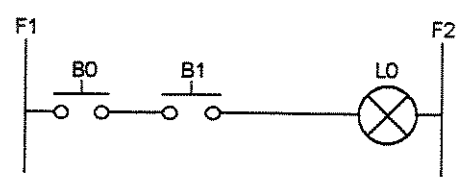


Figura 3.3 – Lógica Convencional – Contatos Elétricos

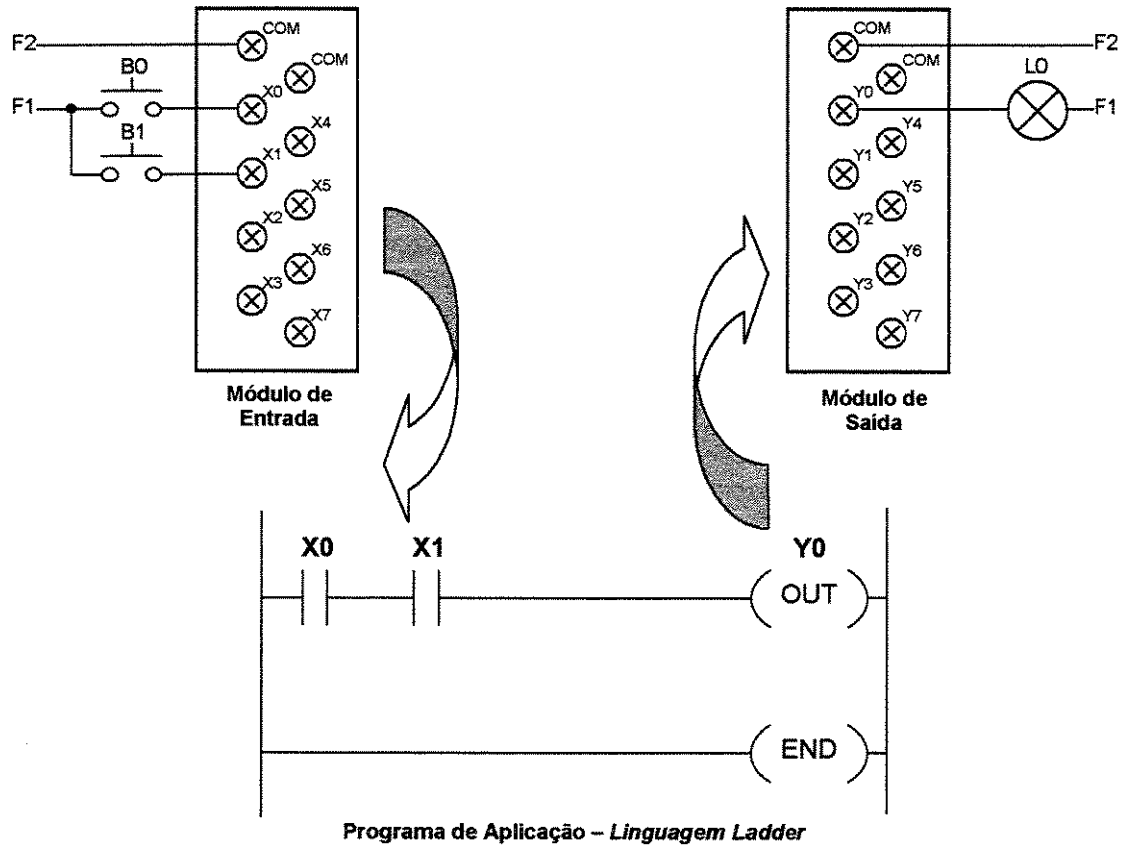


Figura 3.4 – Implementação da Lógica Convencional Através de PLC

3.1 Resumo Histórico

Na década de 60, o aumento da competitividade fez com que a indústria automotiva melhorasse o desempenho de suas linhas de produção, aumentando tanto a qualidade como a produtividade.

Fazia-se necessário encontrar uma alternativa para os sistemas de controle a relés. Uma saída possível, imaginada pela General Motors, seria um sistema baseado no computador.

Assim, em 1968, a Divisão Hydromatic da GM determinou os critérios para projeto do PLC, sendo que o primeiro dispositivo a atender às especificações foi desenvolvido pela Gould Modicon em 1969.

As principais características desejadas nos novos equipamentos de estado sólido, com a flexibilidade dos computadores, eram:

- preço competitivo com os sistemas a relés
- dispositivos de entrada e saída facilmente substituíveis
- funcionamento em ambiente industrial (vibração, calor, poeira, ruídos)
- facilidade de programação e manutenção por técnicos e engenheiros
- repetibilidade de operação e uso

Inicialmente, os PLCs eram chamados PCs - *Programmable Controllers*. Mas, com o advento dos Computadores Pessoais (PCs - *Personal Computers*), convencionou-se PLCs para evitar conflitos de nomenclatura. Originalmente os PLCs foram usados em aplicações de controle discreto (*on/off* – liga/desliga), como os sistemas a relés. Porém, eram facilmente instalados, economizando espaço e energia, além de possuirem indicadores de diagnósticos que facilitavam a manutenção. Uma eventual necessidade de alteração na lógica de controle da máquina era realizada em pouco tempo, através apenas de mudanças no programa, sem necessidade de alteração nas ligações elétricas.

A década de 70 marca uma fase de grande aprimoramento dos PLCs. Com as inovações tecnológicas dos microprocessadores, maior flexibilidade, e um grau também maior de inteligência, foram dadas aos Controladores Lógicos Programáveis:

1972 - Funções de temporização e contagem

1973 - Operações aritméticas, manipulação de dados e comunicação com computadores

1974 - Comunicação com Interfaces Homem-Máquina

1975 - Maior capacidade de memória, controles analógicos, e controle PID

1979/80 - Módulos de I/O remotos, módulos inteligentes e controle de posicionamento

Nos anos 80, aperfeiçoamentos foram atingidos fazendo do PLC um dos equipamentos mais atraentes na Automação Industrial. A possibilidade de comunicação em rede (1981) é hoje uma característica indispensável na indústria. Além dessa evolução tecnológica, foi atingido um alto grau de integração, tanto no número de pontos como no tamanho físico, que possibilitou o fornecimento de minis e micro PLCs (a partir de 1982).

Atualmente, os PLCs apresentam as seguintes características:

- Módulos de I/O de alta densidade (grande número de Pontos de I/O por módulo)
- Módulos remotos controlados por uma mesma CPU
- Módulos inteligentes (co-processadores que permitem realização de tarefas complexas: controle PID, posicionamento de eixos, transmissão via rádio ou modem, leitura de código de barras)
- Softwares de programação em ambiente Windows® (facilidade de programação)
- Integração de Aplicativos Windows® (Access®, Excel®, Visual Basic®) para comunicação com PLCs
- Recursos de monitoramento da execução do programa, diagnósticos e detecção de falhas
- Instruções avançadas que permitem operações complexas (ponto flutuante, funções trigonométricas)
- *Scan Time* (tempo de varredura) reduzido (maior velocidade de processamento) devido à utilização de processadores dedicados
- Processamento paralelo (sistema de redundância) proporcionando confiabilidade na utilização em áreas de segurança
- Pequenos e micro PLCs que oferecem recursos de hardware e software dos PLCs maiores
- Conexão de PLCs em rede (conexão de diferentes PLCs na mesma rede)
- Comunicação através de Rede *Ethernet* (PC é utilizado como CPU, executando a Lógica de Controle)

3.2 Arquitetura Básica dos Controladores Lógicos Programáveis

A seguir, é apresentada a estrutura básica de cada um dos blocos que compõem o PLC.

3.2.1 CPU - Unidade Central de Processamento

O termo CPU está normalmente associado ao processador de um sistema. No entanto, quando se refere ao PLC, determina todos os elementos que formam a 'inteligência' do sistema, compreendendo o processador em si e o sistema de memória, além dos circuitos de controle.

O processador interage continuamente com o sistema de memória para interpretar e executar o Programa de Aplicação (desenvolvido pelo usuário), para controle da máquina ou do

processo. Os circuitos de controle atuam sobre os barramentos de dados (*data bus*), de endereços (*address bus*) e de controle (*control bus*), conforme solicitado pelo processador, similar a um sistema convencional baseado em microprocessador.

Processador

O desenvolvimento tecnológico de um PLC depende principalmente do processador utilizado, que pode ser desde um microprocessador/controlador convencional - 80286, 80386, 8051, até um processador dedicado - *DSP* (*Digital Signal Processor* – Processador Digital de Sinais), por exemplo.

Atualmente, os processadores utilizados em PLCs são dotados de alta capacidade computacional, além daqueles que possuem multi-processamento (vários processadores, com funções específicas, dividem a responsabilidade do controle total) e processamento paralelo, ou sistema de redundância (dois ou mais processadores executam o Programa de Aplicação, confrontando os resultados obtidos após o término de cada execução).

O processador, independente de sua tecnologia, é responsável pelo gerenciamento total do sistema, controlando o barramento de endereços, o barramento de dados e o barramento de controle, além de interpretar e executar as instruções do Programa de Aplicação, podendo operar com registros e palavras de instrução, ou de dados, de diferentes tamanhos (8, 16 ou 32 bits), determinado pelo tamanho de seu acumulador e pela lista de instruções diponíveis para cada PLC.

Sistema de Memória

O Sistema de Memória de um PLC é composto pela Memória do Sistema de Operação (Programa de Execução - *Firmware*, e Rascunho do Sistema) e pela Memória de Aplicação (Programa de Aplicação e Tabela de Dados).

Memória do Sistema de Operação

- *Programa de Execução - Firmware*: Constitui o programa desenvolvido pelo fabricante do PLC, o qual determina como o sistema deve operar, incluindo a execução do Programa de

Aplicação, serviços periféricos, atualização dos Módulos de I/O, etc. O Programa de Execução é responsável pela 'tradução' do Programa de Aplicação desenvolvido pelo usuário – em linguagem de alto nível, para instruções que o microprocessador do PLC possa executar – em linguagem de máquina. É armazenado em memória não-volátil – tipo ROM, normalmente EPROM.

- *Rascunho do Sistema*: Trata-se de uma área de memória reservada para o armazenamento temporário de uma quantidade pequena de dados, utilizados pelo Sistema de Operação para cálculos ou controle (calendário e relógio internos, sinalizadores –*flags*– de alarmes e erros, etc). Uma característica dessa área de memória é o acesso rápido, sendo do tipo RAM.

Memória de Aplicação* ou *Memória do Usuário

- *Programa de Aplicação*: Nesta área é armazenado o programa desenvolvido pelo usuário para execução do controle desejado. Trata-se normalmente de memória EEPROM, podendo ser também EPROM, ou ainda RAM com bateria de segurança.
- *Tabela de Dados*: Esta área armazena dados que são utilizados pelo Programa de Aplicação, como valores atuais e de *preset* (pré-configurado) de temporizadores/ contadores e variáveis do programa, além dos *status* das entradas e saídas (Tabela de Imagem das Entradas e Tabela de Imagem das Saídas) que são lidas e escritas pelo Programa de Aplicação. A atualização desses *status* é realizada constantemente, refletindo as mudanças ocorridas nos Pontos de Entrada, e as atualizações das saídas são efetuadas pelo Programa de Aplicação. Cada Ponto de Entrada e Saída, conectado aos Módulos de I/O, tem um endereço específico na Tabela de Dados, o qual é acessado pelo Programa de Aplicação. Esta memória é do tipo RAM, podendo ser alimentada com bateria de lítio (memória retentiva).

Cada instrução que o PLC pode executar consome uma quantidade pré-determinada de memória, expressa em *bytes* (8 bits) ou *words* (16 bits). Normalmente, as especificações técnicas de um PLC indicam a quantidade de memória disponível para o usuário (memória variável - RAM, e memória de programação – EPROM, EEPROM ou RAM com bateria), podendo ser

expressa em *Kbytes*[2] ('capacidade física' de armazenamento da memória) ou em *Kwords* - palavras de programação ('capacidade lógica' de armazenamento da memória). No entanto, durante a configuração de um PLC deve ser considerada a quantidade de palavras de programação, uma vez que nem sempre há relação direta entre a capacidade física (*Kbytes*) e a capacidade lógica (*Kwords*).

Conforme o fabricante, e a Família (ou modelo) de PLC, a quantidade de memória destinada ao Programa de Aplicação pode ser configurada pelo usuário, ou seja, uma mesma CPU pode ser configurada para aceitar até 2*Kwords* de instruções, como até 4*Kwords*, por exemplo. Normalmente, quando existe esta possibilidade, a memória se apresenta na forma de "cartuchos" que são inseridos na CPU. Existem casos, em que a CPU é fornecida com uma quantidade básica de memória, a qual pode ser expandida através destes "cartuchos".

Além da quantidade de memória, pode haver diferenças na forma de armazenamento dos dados, conforme indicado na tabela 3.1. O Anexo I apresenta uma breve revisão sobre Memórias Semicondutoras.

Sistema de Memória do Plc	
Memória do Sistema de Operação	**Memória de Aplicação**
Programa de Execução *- ROM / EPROM -*	Programa de Aplicação *- RAM (Bateria) / EPROM / EEPROM -*
Rascunho do Sistema *- RAM (Bateria Opcional) -*	Tabela de Dados *- RAM (Bateria Opcional) -*

Tabela 3.1 – Resumo do Sistema de Memória do PLC

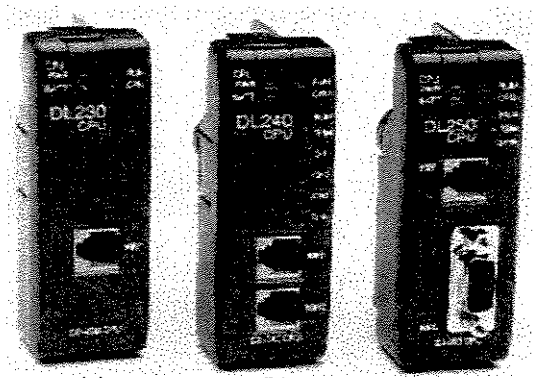
As características normalmente apresentadas nas especificações técnicas de uma CPU e que devem ser consideradas durante a configuração da mesma são:

[2] Quando relacionado a armazenamento/manipulação de dados, $1K = 2^{10} = 1024$. A diferenciação de kilo (k), $1k = 10^3$, é feita pela utilização de caracter maiúsculo (K).

- *Capacidade de memória*: quantidade máxima de memória que a CPU pode conter, sendo indicadas separadamente: *Memória Total Para Programa de Aplicação* e *Memória Total Para Tabela de Dados ou Variáveis*
- *Tipo de memória*: forma de armazenamento do Programa de Aplicação. Algumas CPUs possibilitam a escolha do tipo de memória (EPROM ou EEPROM, por exemplo) para este fim
- *Bateria de backup*: indica se a CPU permite utilização de bateria (de lítio) para manutenção da Tabela de Dados (Dados Retentivos), mesmo sem alimentação
- *Pontos de I/O total*: quantidade máxima de Pontos de I/O que a CPU pode controlar. Conforme o caso, há limites para Pontos de Entrada e Pontos de Saída separadamente. Por exemplo, uma CPU pode controlar 640 Pontos de I/O, tendo no máximo 320 Pontos de Entrada e 320 Pontos de Saída
- *Tempo de processamento* ou *tempo de execução*: tempo necessário para a CPU executar uma instrução booleana (contato). Algumas CPUs podem apresentar tempo de execução para instruções booleanas relativamente alto, por serem indicadas ao processamento de operações mais complexas, como operações aritméticas e trigonométricas. Pode ser expresso em 1k de instruções booleanas, incluindo, ou não, o tempo de *overhead* (processamento executado pela CPU, independente do Programa de Aplicação)
- *Linguagem de programação*: indica a(s) Linguagem(s) de Programação que pode ser utilizada. Apresenta, também, o sistema operacional necessário para o Software de Programação para PC (DOS® ou Windows®, normalmente)
- *Recursos de programação*: indica os principais recursos disponíveis na CPU para programação. Por exemplo, pode apresentar a quantidade de temporizadores e contadores, operação com números inteiros ou números reais (ponto flutuante), rotinas internas para controle PID, existência de calendário/relógio internos, proteção através de senha (para acesso ao programa armazenado na memória) e sistema de diagnósticos, entre outros
- *Portas de comunicação*: quantidade de portas de comunicação existentes na CPU, indicando tipo (RS-232 e/ou RS-422, por exemplo) e protocolos suportados

Para casos em que a CPU se apresenta como um módulo independente, deve-se considerar também o item *potência consumida da base*, o qual especifica a corrente que a CPU consome da

Fonte de Alimentação, através do barramento da Base, para poder operar. Este valor será utilizado no Cálculo de Consumo de Potência, durante a configuração do PLC.



Modelos de CPUs Disponíveis na Mesma Família de PLC – Família DL205, *PLC Direct by Koyo*

	(a) CPU DL230	(b) CPU DL240	(c) CPU DL250
Memória Máxima	2.4K	3.8K	14.5K
Programa de Aplicação	2048	2560	7680
Tabela de Dados	384	1280	7168
Pontos de I/O	128	1152	1152
Tempo de Processamento			
Instrução Booleana	3,3μs	1,4μs	0,61μs
1k Instrução Booleana	4 – 6ms	10 – 12ms	1 – 2ms
Recursos de Programação			
Números Reais	NÃO	NÃO	SIM
Controle PID	NÃO	NÃO	4 Loops
Calendário/Relógio	NÃO	SIM	SIM
Proteção Por Senha	SIM	SIM	SIM
Portas de Comunicação	01	02	02
Tipo	RS-232	RS-232	RS-232 / RS-422
Protocolos	Proprietário Koyo	Proprietário e Rede Koyo	Proprietário e Rede Koyo, MODBUS

Figura 3.5 – Exemplos de CPUs Disponíveis no Mercado

A figura 3.5 apresenta três modelos de CPUs disponíveis em uma mesma Família de PLC, relacionando algumas de suas características.

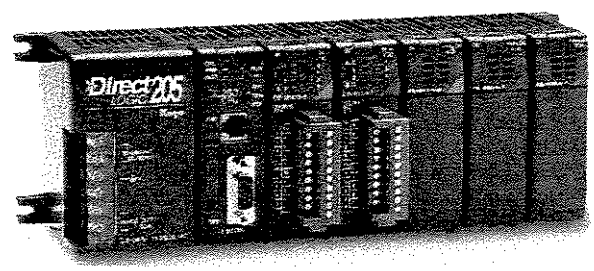
3.2.2 Circuitos/Módulos de I/O

A diferenciação de nomenclatura, Circuitos de I/O ou Módulos de I/O, deve-se ao tipo de PLC. No caso de PLCs Compactos – CPU e I/O alojados em um único invólucro, usa-se Circuitos de I/O. Para PLCs Modulares – CPU e I/O disponíveis de forma independentes, usa-se Módulos de I/O. A partir deste ponto, é usado o termo Módulos de I/O indistintamente.

A figura 3.6 apresenta um PLC Compacto e um PLC Modular.

Os Módulos de I/O fazem a comunicação entre a CPU e o meio externo, além de garantir isolação e proteção a esta. De forma genérica, são divididos em Módulos de Entrada e Módulos de Saída. Para PLCs modulares, há também os Módulos Combinados (Mistos - Pontos de Entrada e Saída no mesmo Módulo).

PLC Compacto
Família DL105, *PLC Direct by Koyo*



PLC Modular
Família DL205, *PLC Direct by Koyo*

Figura 3.6 – Exemplos de PLCs Compacto e Modular Disponíveis no Mercado

Módulos de Entrada (Input Modules)

Recebem os sinais dos dispositivos de entrada tais como sensores, chaves e transdutores, e os convertem em níveis adequados para serem processados pela CPU.

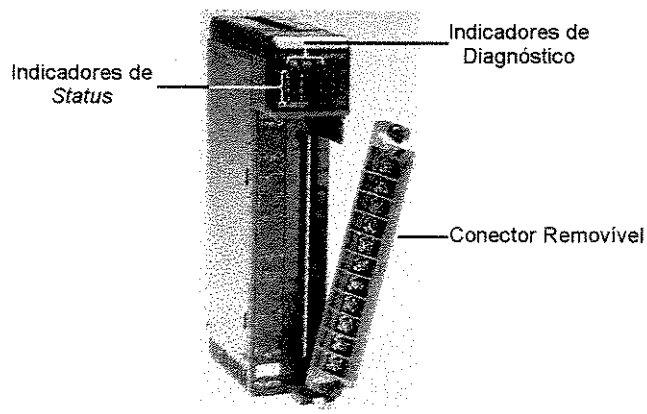
Módulos de Saída (Output Modules)

Enviam os sinais aos dispositivos de saída tais como motores, atuadores e sinalizadores. Estes sinais podem ser resultantes da lógica de controle, pela execução do Programa de Aplicação, ou podem ser 'forçados' pelo usuário, independente da lógica de controle.

Normalmente, os Módulos de I/O são dotados de:

- Isolação Óptica, para proteção da CPU, Fonte de Alimentação e demais Módulos de I/O. Neste caso, não há conexão elétrica entre os dispositivos de entrada (chaves, sensores) e saída (atuadores, motores), e o barramento de comunicação da CPU
- Indicadores de *Status*, para auxílio durante a manutenção. Tratam-se de LEDs (*Ligth Emitting Diodes* – Diodos Emissores de Luz) presentes na parte frontal dos Módulos de I/O que indicam quais Pontos de Entrada estão recebendo sinal dos dispositivos externos, e quais Pontos de Saída estão sendo atuados pela CPU. Há também a possibilidade de existirem indicadores de falhas, como por exemplo falta de alimentação externa, bloco de terminais desconectado, ou fusível queimado
- Conectores Removíveis, que reduzem o tempo de manutenção e/ou substituição dos Módulos de I/O, agilizando tais tarefas

Na figura 3.7 podem ser observadas algumas dessas características.



Módulo de Saída a Transistor – Família DL405, *PLC Direct by Koyo*

Figura 3.7 – Exemplo de Módulo de I/O Disponível no Mercado

Os Módulos de I/O são classificados como Discretos (Digitais) ou Analógicos, existindo também os Especiais em algumas Famílias de PLCs.

Módulos Discretos

Tratam sinais digitais (*on/off* - 0/1). São utilizados em sistemas seqüenciais e na maioria das aplicações com PLCs, mesmo como parte de sistemas contínuos.

Cada Ponto, de Entrada ou Saída, dos Módulos Discretos, corresponde a um bit de um determinado endereço da Tabela de Dados (Tabela de Imagem das Entradas e Tabela de Imagem das Saídas), a qual é acessada durante a execução do Programa de Aplicação, como apresentado na figura 3.8.

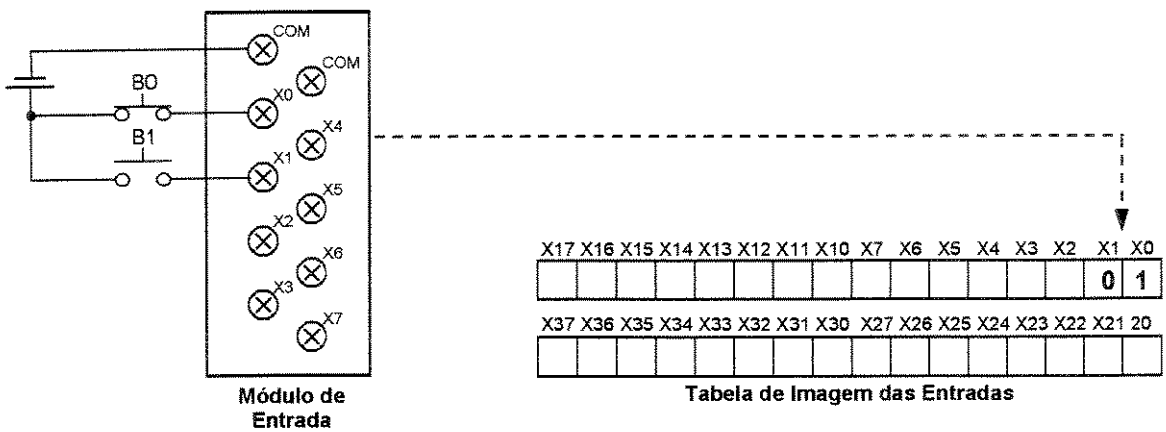


Figura 3.8 – Relação Pontos de Entrada x Tabela de Imagem das Entradas

Os Módulos Discretos de Entrada normalmente apresentam as seguintes características:

- Filtros de sinal, que eliminam problemas de '*bounces*' (pulsos indesejados, causados durante a abertura ou fechamento de contatos mecânicos – "rebatimentos")
- Quantidade de pontos disponíveis: 8, 16, 32 ou 64
- Tipo e faixa de tensão das entradas: AC (110V ou 220V), DC (12V, 24V ou 125V), AC/DC – '*either*' (12V, 24V, 110V), TTL ou 'contato seco'
- As entradas DC podem ter configuração *current sinking* (consumidora de corrente – comum negativo), *current sourcing* (fornecedora de corrente – comum positivo) ou *current sinking/sourcing* (quando possuem um opto-acoplador com dois LEDs em anti-paralelo). Esta é uma cracterística determinante durante a configuração de um PLC, pois dependendo

dos dispositivos de entrada utilizados (sensores NPN ou PNP, por exemplo) faz-se necessário optar por um ou outro tipo de entrada DC

A figura 3.9 apresenta a configuração típica para uma entrada tipo *sinking*. E a figura 3.10 apresenta a configuração típica para uma entrada tipo *sourcing*.

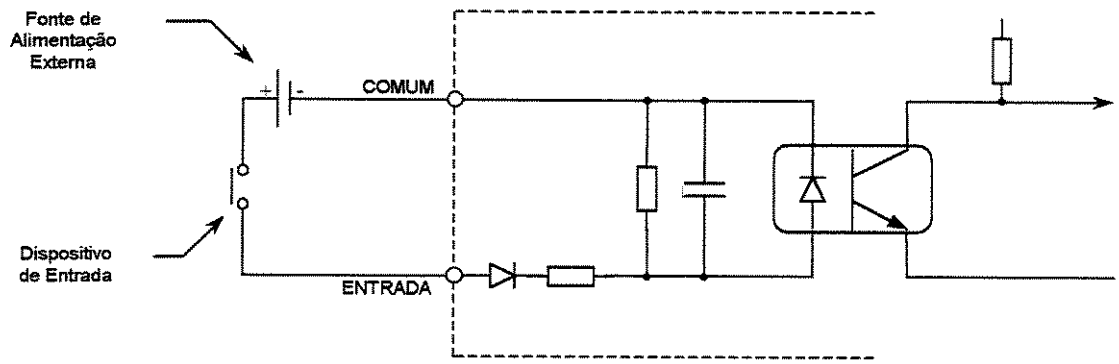


Figura 3.9 – Configuração Típica de uma Entrada Tipo *Sinking* (Comum Negativo)

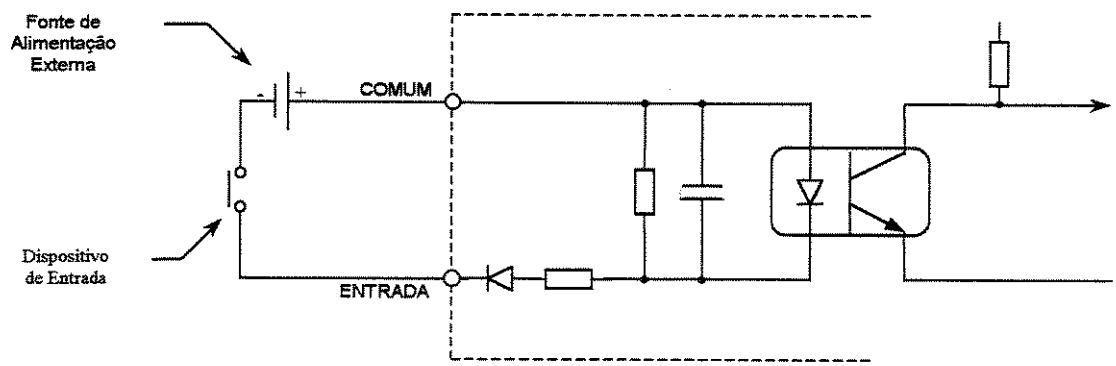


Figura 3.10 – Configuração Típica de uma Entrada Tipo *Sourcing* (Comum Positivo)

Além da *quantidade de pontos*, *tipo* e *tensão das entradas*, os seguintes itens são normalmente apresentados nas especificações técnicas dos Módulos Discretos de Entrada e devem ser considerados durante a configuração dos mesmos:

- *Tensão máxima para nível 0*: máxima tensão permitida para que o Módulo de Entrada reconheça como nível 0 (*off* – desligado)
- *Tensão mínima para nível 1*: mínima tensão necessária para que o Módulo de Entrada reconheça como nível 1 (*on* – ligado)

- *Tensão de pico*: máxima tensão permitida para cada Ponto de Entrada, normalmente com limite de tempo para permanência neste valor
- *Corrente máxima em nível 0*: máxima corrente necessária para que a entrada opere adequadamente em nível 0
- *Corrente mínima em nível 1*: mínima corrente necessária para que a entrada opere adequadamente em nível 1
- *Corrente de entrada*: corrente típica de operação para uma entrada ativa (nível 1)
- *Impedância de entrada*: resistência que cada entrada representa para o dispositivo a ela conectado. Como esta não é linear, deve ser apresentada para algumas faixas de corrente
- *Tempo de resposta de 0 para 1*: tempo (típico) que o módulo necessita para processar a transição de uma entrada, do nível 0 (*off* – desligado) para o nível 1 (*on* – ligado)
- *Tempo de resposta de 1 para 0*: tempo (típico) que o módulo necessita para processar a transição de uma entrada, do nível 1 (*on* – ligado) para o nível 0 (*off* – desligado)
- *Pontos comuns por módulo*: quantidade de 'pontos comuns' disponíveis no módulo, indicando se os mesmos são isolado ou não. Por exemplo, se o Módulo de Entrada for DC, tipo *sinking/sourcing* e possuir dois comuns (A e B) isolados, os Pontos de Entrada relativos ao 'Comum A' podem ser configurados como *sinking* (Comum A conectado ao negativo), e os Pontos de Entrada relativos ao 'Comum B' podem ser configurados como *sourcing* (Comum B conectado ao positivo)
- *Freqüência AC*: freqüência em que o módulo pode operar. Apenas para os Módulos de Entrada AC
- *Potência consumida da base*: especifica a corrente que o módulo consome da Fonte de Alimentação, através do barramento da Base, para poder operar. Este valor será utilizado no Cálculo de Consumo de Potência, durante a configuração do PLC
- *Necessidade de alimentação externa*: alguns módulos, além da fonte externa para fornecimento de tensão às entradas, necessitam de alimentação externa para operarem adequadamente. Na maioria dos casos, estas duas alimentações externas podem ser derivadas da mesma fonte

Os Módulos Discretos de Saída normalmente apresentam as seguintes características:

- Quantidade de pontos disponíveis: 4, 8, 12, 16, 32 ou 64

- Tipo e faixa de tensão das saídas: AC – *triac* ou *scr* (24V, 110V ou 220V), DC – *transistor bipolar* ou *MOS-FET* (5V, 12V, 24V ou 125V) ou relé (AC e DC)
- As saídas DC podem ser tipo *sinking* (consumidora de corrente – comum negativo) ou *sourcing* (fornecedora de corrente – comum positivo)
- As saídas a relé podem ter contatos simples (um contato normalmente aberto), ou reversíveis (um contato normalmente aberto e outro normalmente fechado)

A figura 3.11 apresenta a configuração típica para uma saída tipo *sinking*.

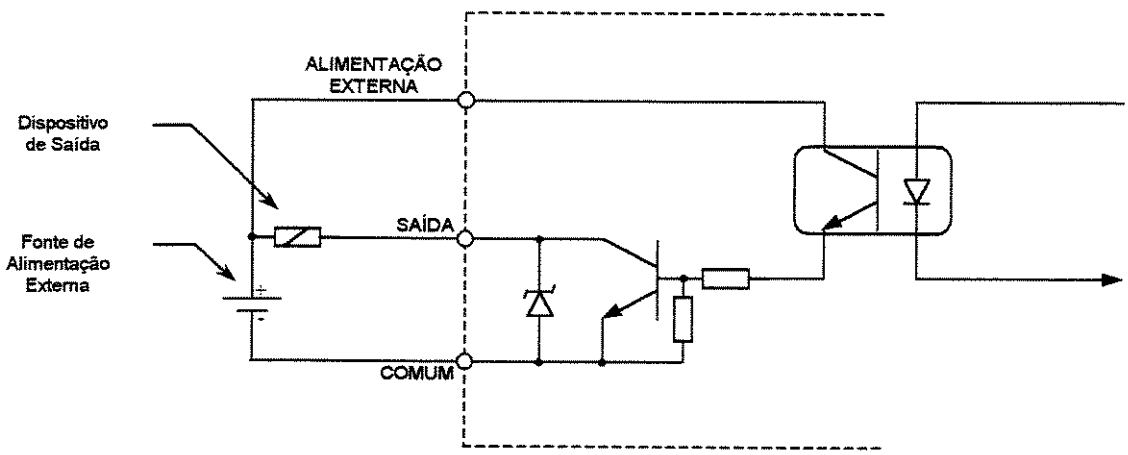


Figura 3.11 – Configuração Típica de uma Saída Tipo *Sinking* (Comum Negativo), sem Fusível de Proteção

Além da *quantidade de pontos*, *tipo* e *tensão das saídas*, os seguintes itens são normalmente apresentados nas especificações técnicas dos Módulos Discretos de Saída e devem ser considerados durante a configuração dos mesmos:

- *Tensão de pico*: tensão máxima permitida para cada Ponto de Saída, normalmente com limite de tempo para permanência neste valor
- *Queda de tensão*: também denominada "tensão de saturação", indica a tensão medida entre um Ponto de Saída, quando acionado, e o comum, com carga máxima
- *Corrente máxima*: máxima corrente permitida para cada Ponto de Saída, normalmente indicada para cargas resistivas. Atenção especial deve ser dada a este item, pois na maioria dos casos, são indicadas *corrente máxima/ponto* e *corrente máxima/comum* ou *máxima/módulo*. Por exemplo, um módulo com 8 Pontos de Saída pode ter a seguinte

indicação de *corrente máxima*: 1A/ponto e 5A/comum, indicando que cada ponto individualmente pode acionar uma carga de até 1A, e a somatória da corrente de todos os pontos acionados não deve exceder os 5A

- *Corrente de pico*: máxima corrente que pode ser fornecida à carga por um curto intervalo de tempo, durante a transição de 0 para 1. Este valor é maior que o de *corrente máxima* e característico para acionamento de circuitos indutivos
- *Corrente de fuga*: máxima corrente que poderá circular pelo dispositivo de saída com o Ponto de Saída não acionado (*off* – desligado)
- *Carga mínima*: menor corrente que o Ponto de Saída deve fornecer à carga para operar adequadamente
- *Tempo de resposta de 0 para 1*: tempo (típico) que o módulo necessita para processar a transição de uma saída, do nível 0 (*off* – desligado) para o nível 1 (*on* – ligado)
- *Tempo de resposta de 1 para 0*: tempo (típico) que o módulo necessita para processar a transição de uma saída, do nível 1 (*on* – ligado) para o nível 0 (*off* – desligado)
- *Pontos comuns por módulo*: quantidade de 'pontos comuns' disponíveis no módulo, indicando se os mesmos são isolado ou não. Por exemplo, se for um Módulo de Saída a Relé, e possuir dois comuns (A e B) isolados, os Pontos de Saída relativos ao 'Comum A' podem ser configurados para operarem com tensão DC, e os Pontos de Saída relativos ao 'Comum B' podem ser configurados operarem com tensão AC
- *Freqüência AC*: freqüência em que o módulo pode operar. Apenas para os Módulos de Saída AC e Relé
- *Potência consumida da base*: especifica a corrente que o módulo consome da Fonte de Alimentação, através do barramento da Base, para poder operar
- *Necessidade de alimentação externa*: alguns módulos, além da fonte externa para fornecimento de tensão às saídas, necessitam de alimentação externa para operarem adequadamente
- *Fusíveis de proteção*: indica a existência ou não destes elementos, se os mesmos são substituíveis, e se estão localizados interna ou externamente ao módulo. Mesmo que os Módulos de Saída apresentem *fusíveis de proteção*, recomenda-se a utilização de proteção externa, através de fusíveis individuais para cada Ponto de Saída

Outro fator importante durante a configuração dos Módulos de Saída, relaciona-se ao acionamento dos dispositivos controlados. Não é recomendada a utilização de saídas a relé para *acionamentos cíclicos*, mesmo de baixa freqüência, e *acionamentos rápidos*, devido a fadiga mecânica que os mesmos podem sofrer.

Porém, quando se utiliza saídas a relé para acionamento de cargas indutivas, recomenda-se a utilização de circuito RC – *snubber* (AC e DC) e diodo (apenas DC) para proteção dos contatos.

A figura figura 3.12 apresenta as proteções recomendadas para as saídas a relés; e as equações 3.1 e 3.2 apresentam o cálculo para o capacitor e o resistor do circuito *snubber*, respectivamente.

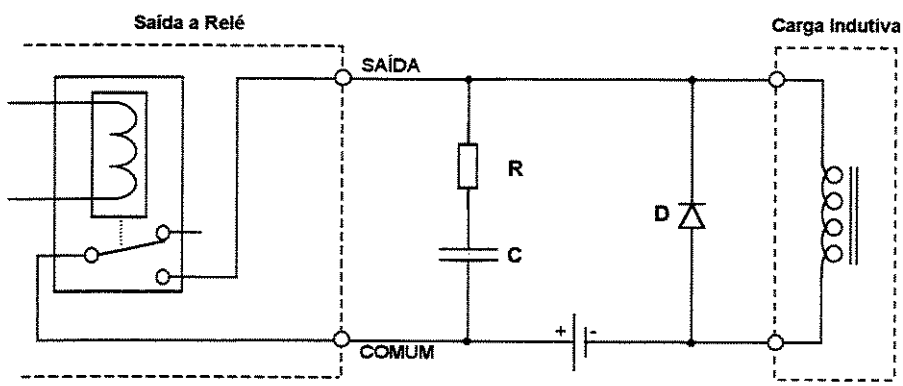


Figura 3.12 – Proteções Recomendadas para Saídas a Relé

$$C\ (\mu F) = \frac{I^2}{10}$$

Equação 3.1 – Cálculo de Capacitor (*Snubber*)

$$R\ (\Omega) = \frac{V}{10 \cdot I^x}, \text{ onde } x = 1 + \frac{50}{V}$$

Equação 3.2 – Cálculo de Resistor (*Snubber*)

V = tensão sobre o contato, quando aberto. Para AC, $V = V_{pico} = V_{rms} \cdot 1{,}414$

I = corrente que circula pelo contato, quando fechado. Para AC, $I = I_{pico} = I_{rms} \cdot 1{,}414$

$\mathbf{R_{mínimo}} = 0{,}5\Omega$ ±5%, 1/2W

$\mathbf{C_{mínimo}} = 0{,}001\mu F$ e tensão ≥ 150% V, não polarizado (poliéster metalizado)

Diodo deve suportar tensão reversa de pelo menos 100V e corrente direta de 3A. O tipo mais indicado é o *schottky*, devido à alta velocidade de comutação.

A quantidade de pontos de um módulo determina sua densidade, sendo que para os Módulos de Saída, quanto maior a densidade, menor a capacidade de corrente que cada ponto pode fornecer.

Módulos Analógicos

Tratam sinais analógicos (tensão, corrente, temperatura, por exemplo). São utilizados em sistemas contínuos ou como parte de sistemas seqüenciais.

Os Módulos Analógicos de Entrada convertem sinais analógicos, provenientes dos dispositivos de entrada (transdutor, conversor, termopar), em sinais digitais por meio de Conversor Analógico/Digital (*ADC – Analog to Digital Converter*), disponibilizando-os adequadamente ao barramento da CPU.

Os Módulos Analógicos de Saída convertem sinais digitais, disponíveis no barramento da CPU, em sinais analógicos por meio de Conversor Digital/Analógico (*DAC – Digital to Analog Converter*), enviando-os aos dispositivos de saída (driver, amplificador).

Cada entrada ou saída analógica é denominada de canal, em vez de ponto como nos Módulos Discretos. O valor convertido referente a cada canal analógico de entrada, ou o valor a ser convertido e enviado para cada canal analógico de saída, é armazenado em um endereço específico na Tabela de Dados, determinado pelo Programa de Aplicação, e a quantidade de *bits* relativos a cada canal depende da resolução dos Conversores A/D e D/A.

A figura 3.13 exemplifica a relação "Canal de Entrada x Tabela de Dados" para um Módulo Analógico.

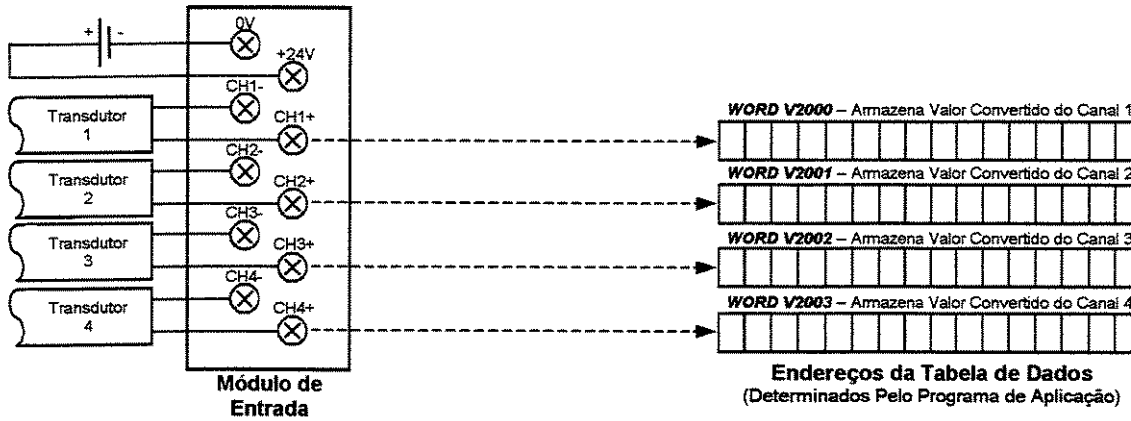


Figura 3.13 – Relação Canal de Entrada x Tabela de Dados

Independente da quantidade de canais disponíveis e da resolução dos conversores, cada Módulo Analógico de Entrada ou Saída, consome uma certa quantidade de Pontos de I/O, a qual tem variações de acordo com a Família de PLC utilizada.

Os Módulos Analógicos de Entrada normalmente apresentam as seguintes características:

- Filtro ativo, para eliminação de possíveis ruídos presentes nos sinais de entrada
- Alta impedância de entrada, para os canais com faixas de operação em tensão, que possibilita conexão a uma vasta gama de dispositivos, eliminando problemas de incompatibilidade de sinais
- Multiplexador para os canais de entrada, que determina o canal a ser enviado ao Conversor Analógico/Digital
- Processador dedicado, responsável pelo processamento e precisão do sinal digital enviado à CPU, além diagnósticos referentes ao módulo
- Quantidade de canais disponíveis: 2, 4, 8 ou 16
- Tipo e faixa de operação dos canais: corrente (0-20mA, 4-20mA), tensão (0-5V, ±5V, 0-10V, ±10V) ou temperatura (termopares – J, E, K, R, S, T, B, , ou termo-resistências – Pt100, jPt100, Pt1000, Cu10, Cu25)
- Um mesmo módulo pode operar em mais de uma faixa, a qual é selecionada por *dip-switches* ou *jumpers* internos ao módulo. Por exemplo, um Módulo Analógico de 04 Canais de Entrada, tipo tensão, pode operar tanto na faixa de 0-5V como nas faixas de ±5V, 0-10V

ou ±10V. Esta seleção pode ser individual para cada canal, ou seja, canal 1 opera na faixa de 0-5V, canais 2 e 3 operam na faixa de ±5V e canal 4 opera na faixa de 0-10V, por exemplo; ou ser única para todos os canais, quando canais 1 a 4 operam na mesma faixa selecionada. Há ainda a possibilidade de um mesmo módulo operar tanto em faixas de corrente como em faixas de tensão, sendo também selecionadas por *dip-switches* ou *jumpers*

A figura 3.14 apresenta a arquitetura básica de um Módulo Analógico de Entrada, com quatro canais.

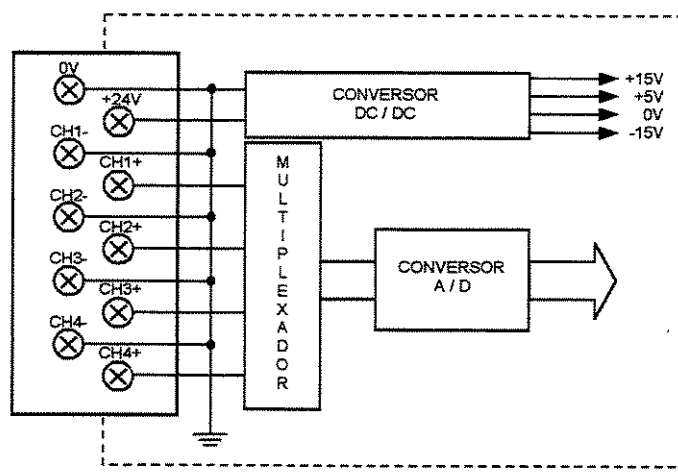


Figura 3.14 – Arquitetura Básica de um Módulo Analógico de Entrada

As características normalmente apresentadas nas especificações técnicas dos Módulos Analógicos de Entrada e que devem ser consideradas durante a configuração dos mesmos são:

- *Isolação dos canais: isolados* (isolação galvânica) - que possibilita conexão a dispositivos com saída diferencial (os dois Pontos de Entrada variam em relação ao terra – *ground*, e o valor a ser convertido é a diferença existente entre estes dois pontos), ou *não-isolados* (*comuns*) – um dos Pontos de Entrada é conectado ao terra da fonte, internamente
- *Resolução*: menor incremento possível no valor analógico de entrada que pode ser detectado pelo Conversor Analógico/Digital, normalmente expresso em *bits*. Por exemplo, um Módulo Analógico de Entrada, configurado para operar em tensão, na faixa de 0-10V, tem 12bits de resolução. Como $2^{12} = 4096$ (0 a 4095), a faixa de operação do módulo (0-

10V) é dividida em 4095 frações, ou seja, cada fração é igual a 2,44mV, sendo esta a *resolução* do módulo. Quanto maior a quantidade de *bits* do Conversor Analógico/Digital, menor o incremento possível e melhor a *resolução*. A maioria dos módulos encontrados no mercado apresenta *resolução* de 12bits

- *Tipo de conversão*: método utilizado para converter o sinal analógico em sinal digital, sendo na maioria dos casos por aproximação sucessiva
- *Razão de atualização*: 'tempo' necessário para que os sinais analógicos sejam digitalizados e reconhecidos pela CPU, normalmente expresso em *canal(s)/scan*. Por exemplo, um Módulo Analógico de 04 Entradas, pode apresentar *razão de atualização* de 1 canal/*scan* (ciclo de processamento do PLC) para uma determinada CPU e de 4 canais/*scan* para outra CPU mais sofisticada. Ou seja, para a primeira CPU seriam necessários 4 *scans* para leitura de todos os canais do módulo. O mesmo é conseguido em apenas um *scan* com a utilização da segunda CPU
- *Erro de linearidade*: precisão relativa da representação digital sobre a faixa de operação do sinal de entrada. Pode ser expressa em *bits* ou em porcentagem
- *Erro máximo*: erro máximo absoluto que pode haver entre a representação digital e o valor do sinal analógico existente na entrada do canal. Os fatores que contribuem para este erro (erro na calibração de fundo de escala, erro na calibração de *off-set* e influência da temperatura) são especificados separadamente
- *Pontos de I/O consumidos*: quantidade de Pontos de Entrada que o módulo consome da CPU
- *Potência consumida da base*: especifica a corrente que o módulo consome da Fonte de Alimentação, através do barramento da Base, para poder operar. Este valor será utilizado no Cálculo de Consumo de Potência, durante a configuração do PLC
- *Fonte de alimentação externa*: indica a tensão da fonte externa, sendo de 12VDC ou 24VDC na maioria dos casos

Os Módulos Analógicos de Saída normalmente apresentam as seguintes características:

- Quantidade de canais disponíveis: 2, 4, 8 ou 16
- Tipo e faixa de operação dos canais: corrente (0-20mA, 4-20mA) ou tensão (0-5V, ±5V, 0-10V, ±10V)

- Um mesmo módulo pode operar em mais de uma faixa, a qual é selecionada por *dip-switches* ou *jumpers* internos ao módulo

A figura 3.15 apresenta a arquitetuta básica de um Módulo Analógico de Saída, com dois canais.

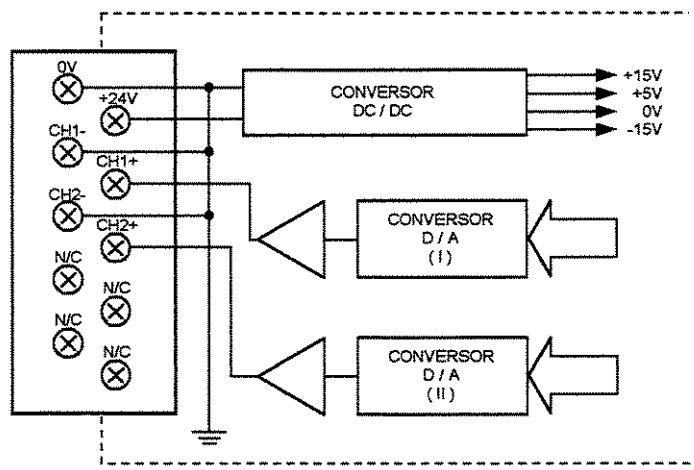


Figura 3.15 – Arquitetura Básica de um Módulo Analógico de Saída

As características normalmente apresentadas nas especificações técnicas dos Módulos Analógicos de Saída e que devem ser consideradas durante a configuração dos mesmos são:

- *Tipo dos canais: isolados* (isolação galvânica) ou *não-isolados* (*comuns*)
- *Impedância de saída*: apresenta as resistências mínima e máxima a que o canal de saída possa ser conectado, para sinais de corrente e tensão específicos
- *Resolução*: o menor incremento que o dado enviado ao Conversor Digital/Analógico pode causar no valor analógico de saída, normalmente expresso em *bits*. A maioria dos módulos encontrados no mercado apresenta *resolução* de 12bits
- *Razão de atualização*: 'tempo' necessário para que os valores digitais provenientes da CPU sejam convertidos em sinais analógicos para as saídas, normalmente expresso em *canal(s)/scan*
- *Erro de linearidade*: precisão relativa da representação digital sobre a faixa de operação do sinal de saída. Pode ser expressa em *bits* ou em porcentagem

- *Erro máximo*: erro máximo absoluto que pode haver entre a representação digital e o valor analógico disponível na saída do canal. Os fatores que contribuem para este erro (erro na calibração de fundo de escala, erro na calibração de *off-set* e influência da temperatura) são especificados separadamente
- *Pontos de I/O consumidos*: quantidade de Pontos de Saída que o módulo consome da CPU
- *Potência consumida da base*: especifica a corrente que o módulo consome da Fonte de Alimentação, através do barramento da Base, para poder operar. Este valor será utilizado no Cálculo de Consumo de Potência, durante a configuração do PLC
- *Fonte de alimentação externa*: indica necessidade de alimentação externa, podendo ser de 12VDC ou 24VDC conforme o caso

3.2.3 Fonte de Alimentação

A Fonte de Alimentação desempenha importante papel na operação do sistema de um PLC. Além de fornecer todos os níveis de tensão para alimentação da CPU e dos Módulos de I/O, funciona como um dispositivo de proteção. Garante a segurança e a integridade da tensão de alimentação para todo o sistema, através do monitoramento constante dos níveis de tensão e corrente fornecidos. Se esses níveis excederem os valores máximo ou mínimo permitidos, além do tempo especificado pelo fabricante, a fonte interage diretamente com o processador gerando uma interrupção, através de uma seqüência de comandos, fazendo com que a CPU pare a execução do Programa de Aplicação.

Atualmente, as Fontes de Alimentação dos PLCs utilizam tecnologia de chaveamento de freqüência (fontes chaveadas), sendo que, em alguns casos, a tensão de entrada não é fixa e nem ajustável pelo usuário, possuindo ajuste automático, proporcionando maior versatilidade e qualidade ao sistema. Há, também, Fontes de Alimentação com tensão de entrada DC (12V, 24V ou 125V), para aplicações específicas (automotivas, por exemplo).

As proteções externas recomendadas para a Fonte de Alimentação dos PLCs variam conforme o fabricante, mas basicamente consistem em transformadores de isolação ou supressores de ruídos para rede, aterramento adequado, e conformidade com as normas técnicas locais.

Em alguns casos, os Módulos de I/O necessitam, além das tensões fornecidas pela fonte do PLC, de alimentação externa. A fonte do PLC é responsável pela alimentação do circuito lógico dos Módulos de I/O, sendo que a fonte externa alimenta os circuitos de potência, ou circuitos externos - entrada ou saída (Módulos Discretos e Analógicos) ou ainda, fornece um nível de tensão com maior capacidade de corrente para os Módulos Especiais.

Normalmente, as fontes dos PLCs proporcionam saída auxiliar de tensão em 24VDC, porém com limite reduzido de corrente, na faixa de 300mA a 800mA. Esta saída pode ser utilizada para alimentação dos Módulos de I/O, desde que respeitado o limite de corrente.

A Fonte de Alimentação tem aspectos variados, conforme o fabricante e a Família de PLC. Pode apresentar-se em conjunto com a CPU, ou como um Módulo independente para ser conectado à Base, ou ainda ser parte integrante da própria Base, como apresentado na figura 3.16.

a- **Fonte e CPU como único Módulo** – Família DL405, *PLC Direct by Koyo*
b- **Fonte acoplada à Base** – Família DL205, *PLC Direct by Koyo*
c- **Fonte e CPU como Módulos independentes** – Família SLC500, *Allen-Bradley*

Figura 3.16 – Fontes de Alimentação

As carcaterísticas normalmente apresentadas nas especificações técnicas de uma Fonte de Alimentação e que devem ser consideradas durante a configuração da mesma são:

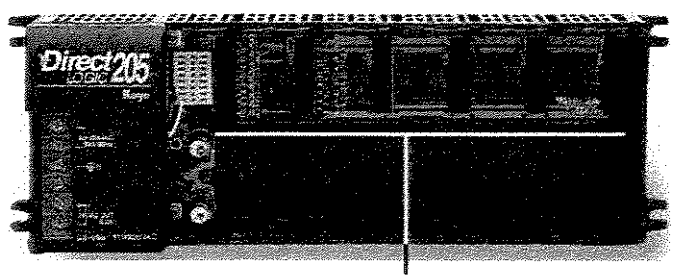
- *Faixa da tensão de entrada*: AC (85-132V, 170-264V, 85-264V, por exemplo), DC (12V, 24V, 10-28V, 125V, por exemplo). Para as faixas de entrada em tensão DC, observar tmbém o *ripple* máximo permitido, geralmente menor que 10%

- *Seleção da faixa de entrada*: automática, por *jumpers*, ou por terminais de conexão
- *Potência fornecida*: máxima corrente fornecida ao barramento da Base, normalmente relacionada à tensão de 5VDC, para alimentação dos Módulos de I/O e da CPU, se for o caso (CPU como módulo independente). Este valor será utilizado no Cálculo de Consumo de Potência durante a configuração do PLC
- *Saída auxiliar de 24VDC*: apresenta as características (tensão, corrente e *ripple*) da saída auxiliar de 24VDC. Apenas para fontes com alimentação AC

3.2.4 Base ou *Rack*

Responsável pela sustentação mecânica dos elementos que compõem o PLC, contém o barramento que faz a conexão elétrica entre os mesmos, no qual estão presentes os sinais de dados, endereço e controle – necessários para comunicação entre a CPU e os Módulos de I/O, além dos níveis de tensão fornecidos pela Fonte de Alimentação – necessários para que a CPU e os Módulos de I/O possam operar.

A figura 3.17 apresenta um exemplo de Base, com indicação do barramento interno.



Base Com Fonte de Alimentação – Família DL205, *PLC Direct by Koyo*

Figura 3.17 – Exemplo de Base Disponível no Mercado

Cada posição da Base, possível de receber um Módulo de I/O ou a CPU – quando esta se apresentar como módulo independente, é denominada de *slot* (ranhura, abertura). E cada *slot* da Base tem uma identificação própria, conforme o fabricante. Por exemplo, a *PLC Direct by Koyo* utiliza a seguinte nomenclatura para os *slots* da Base:

Nas Famílias em que a CPU apresenta-se como um módulo independente (Famílias DL205 e DL305), o primeiro *slot* ao lado da Fonte de Alimentação, denomina-se *slot da CPU*, não podendo ser ocupado por Módulos de I/O. Em casos específicos, de controle baseado em PC, pode ser ocupado por Módulos Especiais de Comunicação (*ethernet*, por exemplo). O primeiro *slot* ao lado da CPU, denomina-se *slot 0*, o seguinte *slot 1*, e assim sucessivamente, conforme apresentado na figura 3.18.

Fonte de Alimentação	SLOT DA CPU	SLOT 0	SLOT 1	SLOT 2	SLOT 3	SLOT 4

Figura 3.18 – Exemplo de Identificação dos *Slots* da Base

Alguns Módulos de I/O ou Especiais podem ter restrições quanto ao posicionamento nos *slots* da Base. Porém, de forma geral, os Módulos Discretos e Analógicos podem ser posicionados livremente pelo usuário. As possíveis restrições de posicionamento são indicadas nos respectivos manuais técnicos.

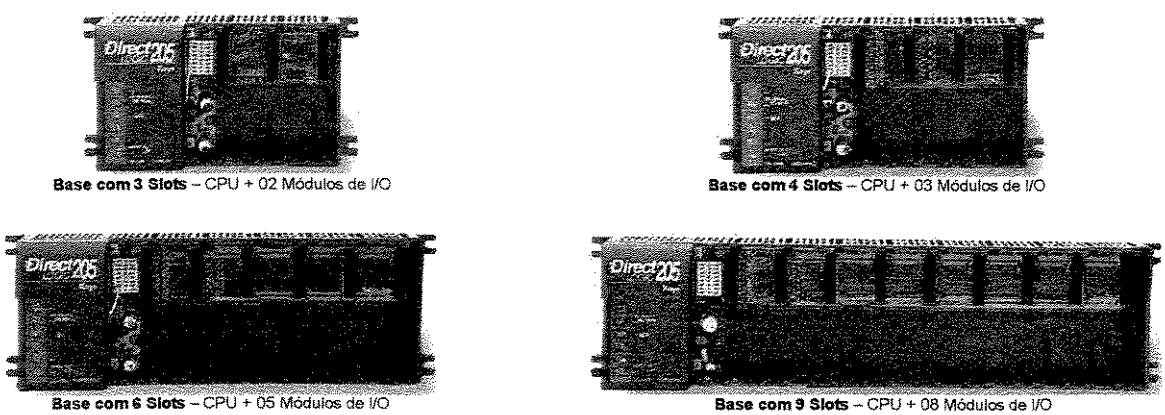


Figura 3.19 – Modelos de Bases

Na maioria dos casos, uma mesma Família de PLC contém Bases com diferentes quantidades de *slots*, com o objetivo de atender às necessidades específicas de cada aplicação. A figura 3.19 exemplifica esta situação.

3.3 Classificação dos Controladores Lógicos Programáveis

Os PLCs podem ser classificados de acordo com a quantidade de Pontos de I/O que a CPU pode controlar e a quantidade de memória de programação disponívesl (embora existam divergências entre autores e fabricantes quanto a esta classificação):

- *Micro PLCs* (até 64 Pontos de I/O e até 2*Kwords* de memória)
- *Pequenos PLCs* (de 64 a 512 Pontos de I/O e até 4*Kwords* de memória)
- *PLCs Médios* (de 256 a 2048 Pontos de I/O e dezenas de *Kwords* de memória)
- *PLCs Grandes* (acima de 2048 Pontos de I/O e centenas de *Kwords* de memória)

A partir de 1997, alguns fabricantes lançaram no mercado PLCs com até 14 Pontos de I/O e tamanho muito reduzido, denominando-os *Nano PLCs*.

Entre os Micro e Pequenos PLCs, ainda é possível encontrar outra divisão:

- *PLCs Compactos*: que têm quantidade fixa de Pontos de I/O
- *PLCs Modulares*: que permitem a configuração, por parte do usuário, da quantidade e combinação dos Pontos de I/O.

Em alguns PLCs Compactos é possível a adição de Pontos de I/O através de 'blocos' de expansão, com limite determinado pelo fabricante. Porém, apresentam poucas opções de configuração (quantidade e tipo dos Pontos de I/O para cada bloco de expansão).

3.4 Configuração dos Controladores Lógicos Programáveis

Além das especificações técnicas da CPU, dos Módulos de I/O, da Fonte de Alimentação e da Base, outros fatores importantes devem ser obervados durante a configuração de PLCs. Como, por exemplo, Configuração do Sistema de I/O, Cálculo de Consumo de Potência e Configuração da Porta Serial.

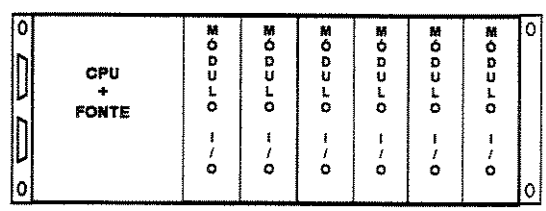
3.4.1 Configuração do Sistema de I/O

A disposição dos Módulos de I/O na(s) Base(s) do PLC está diretamente ligada à Configuração do Sistema de I/O.

Em alguns casos, uma única base (Base Local) não é suficiente para acomodar todos os Módulos de I/O necessários à determinada aplicação, tanto por limitação de espaço físico (quantidade de *slots* disponíveis < quantidade de Módulos de I/O) como por limitação elétrica (corrente fornecida ao barramento < corrente consumida pelos Módulos de I/O). Então, faz-se necessário a utilização de Bases de Expansão Locais. Há também situações, em que os dispositivos de entrada e/ou saída estão localizados distantes da CPU (Base Local), sendo necessário a utilização de Bases de Expansão Remotas.

Base Local

A Base na qual a CPU está instalada é denominada Base Local. Os Módulos de I/O instalados na mesma base que a CPU, são denominados Módulos de I/O Locais. A utilização apenas de Base Local atende à maioria das aplicações com PLCs. A figura 3.20 apresenta a configuração do Sistema de I/O com Base Local apenas.



Base Local - Família DL405, *PLC Direct by Koyo*

Figura 3.20 – Configuração do Sistema de I/O - *Base Local*

A forma de identificação dos Pontos de I/O instalados na Base Local depende da Família de PLC utilizada. Por exemplo, as CPUs *PLC Direct by Koyo* identificam os Pontos de Entrada por 'X', seguido pelo endereço a que se referem, em base octal (X0, X1,..., X7, X10, X11,...). Os Pontos de Saída são identificados por 'Y', seguido pelo endereço, também em base octal (Y0, Y1,..., Y7, Y10, Y11, ...). Outros exemplos de identificação encontrados no mercado são: %I

(*input*), I, E (entrada), para os Pontos de Entrada; %Q (*quit*), O (*output*), S (saída), para os Pontos de Saída; com o endereço expresso em base octal, decimal ou hexadecimal.

O endereçamento dos Pontos de I/O (o que determina que um Ponto de Entrada seja identificado por X0, e não por X10, por exemplo) também depende da Família de PLC utilizada. Algumas associam o endereço ao *slot* em que o Módulo de I/O está instalado; outras, pela seqüência em que os Módulos de I/O aparecem instalados, independente do *slot*.

Normalmente, são encontrados dois métodos de Configuração dos Pontos de I/O, conforme a CPU utilizada:

- Configuração Manual, na qual o endereço dos Pontos de I/O é determinado pelo usuário, por hardware (através de *jumpers* ou *dip-switches* existentes nos módulos) ou por software (através de parâmetros específicos), respeitando alguns critérios
- Configuração Automática, a qual é realizada pela CPU sem intervenção do usuário

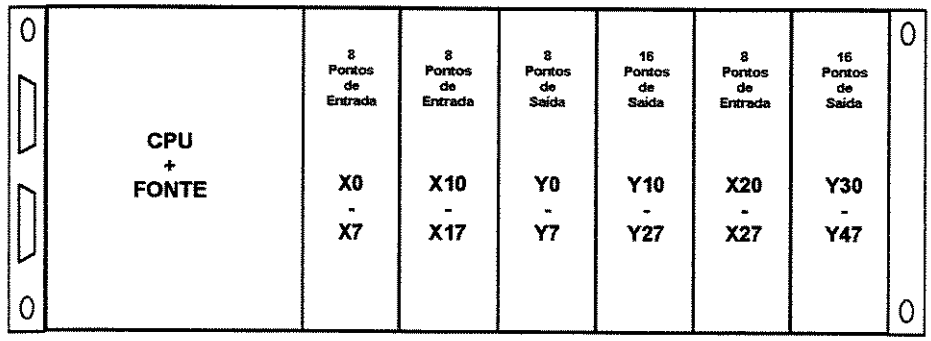
Por exemplo, todas as CPUs *PLC Direct by Koyo* utilizam o método de Configuração Automática e apenas algumas permitem, também, a utilização de Configuração Manual. A Configuração Automática, para este caso, é realizada da seguinte forma:

1- Durante o *power-up* (energização inicial) do PLC, a CPU automaticamente detecta e identifica os Módulos de I/O instalados em cada *slot* da Base

2- O endereçamento é realizado seqüencialmente, a partir do *slot 0*, sendo que o primeiro Ponto de Entrada encontrado é endereçado como X0, o primeiro Ponto de Saída encontrado é endereçado como Y0, e assim sucessivamente para ambos os casos. Os endereços são utilizados em grupos de 8, 16, 32 ou 64 pontos, conforme a densidade dos Módulos de I/O. Por exemplo, um Módulo de 12 Pontos de Saída a Relé, instalado no *slot 0*, utiliza os endereços Y0 – Y17, ou seja, consome 16 Pontos de I/O sendo que apenas 12 existem fisicamente (Y0 – Y5 e Y10 – Y15). O mesmo sistema de endereçamento é utilizado para os Módulos Analógicos e Especiais, sendo que a quantidade de Pontos de I/O consumida é indicada nos respectivos manuais técnicos.

3- O endereçamento é armazenado na memória da CPU (Rascunho do Sistema)

A figura 3.21 demonstra o método de Configuração Automática realizada pelas CPUs *PLC Direct by Koyo*.



Configuração Automática dos Pontos de I/O - Família DL405, *PLC Direct by Koyo*

Figura 3.21 – Exemplo de Configuração Automática do Sistema de I/O

Expansão Local

Esta configuração é utilizada quando se necessita de Pontos de I/O em quantidade superior à que a Base Local pode suportar, ou quando a Fonte de Alimentação Local não é suficiente para fornecer corrente a todos os Módulos de I/O utilizados em determinada aplicação.

O acréscimo, ou distribuição, dos Módulos de I/O é feito através de Base(s) de Expansão situada(s) localmente - próxima(s) à Base Local. As Famílias de PLCs que permitem esta configuração, especificam a quantidade limite de Bases de Expansão e de Pontos de I/O possíveis para cada modelo de CPU. A figura 3.22 exemplifica esta situação.

Na(s) Base(s) de Expansão não há CPU, apenas a Fonte de Alimentação. A comunicação com a Base Local é feita através de Cabo de Conexão apropriado, fornecido pelo fabricante. Os Pontos de I/O dos módulos instalados nas Bases de Expansão Locais são tratados e endereçados pela CPU como os Pontos de I/O dos Módulos instalados na Base Local, sendo atualizados a cada *scan*. Ou seja, as Bases de Expansão Locais são vistas como prolongamento (expansão) dos barramentos de dados, de endereços e de controle da Base Local, com alimentação própria.

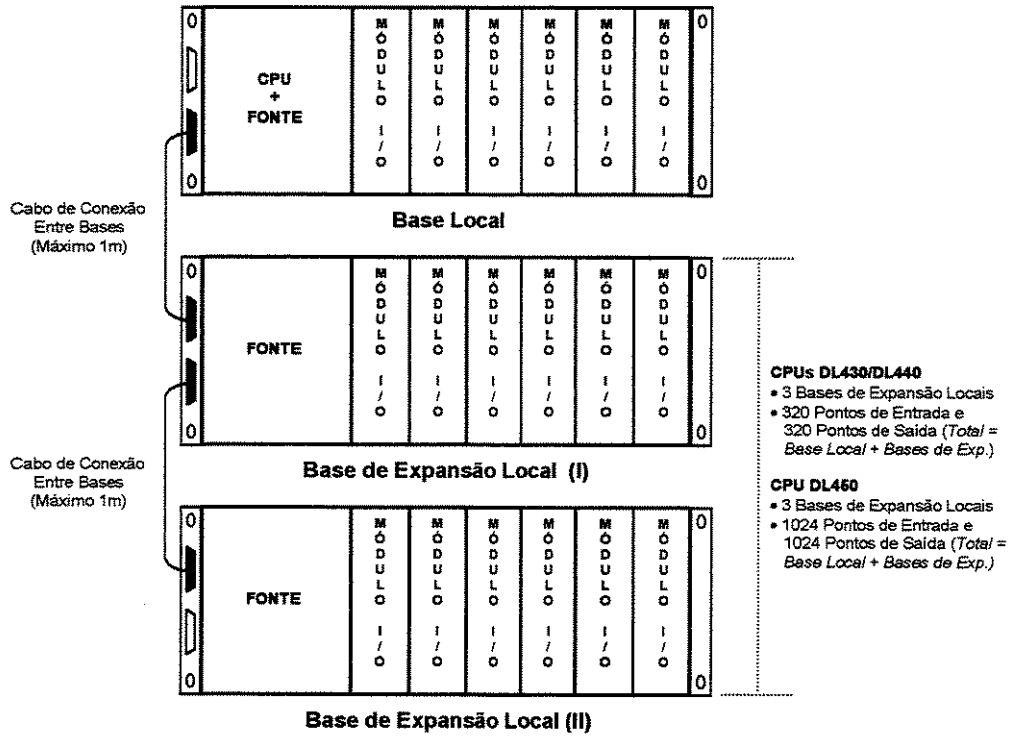


Figura 3.22 – Configuração do Sistema de I/O - *Expansão Local*

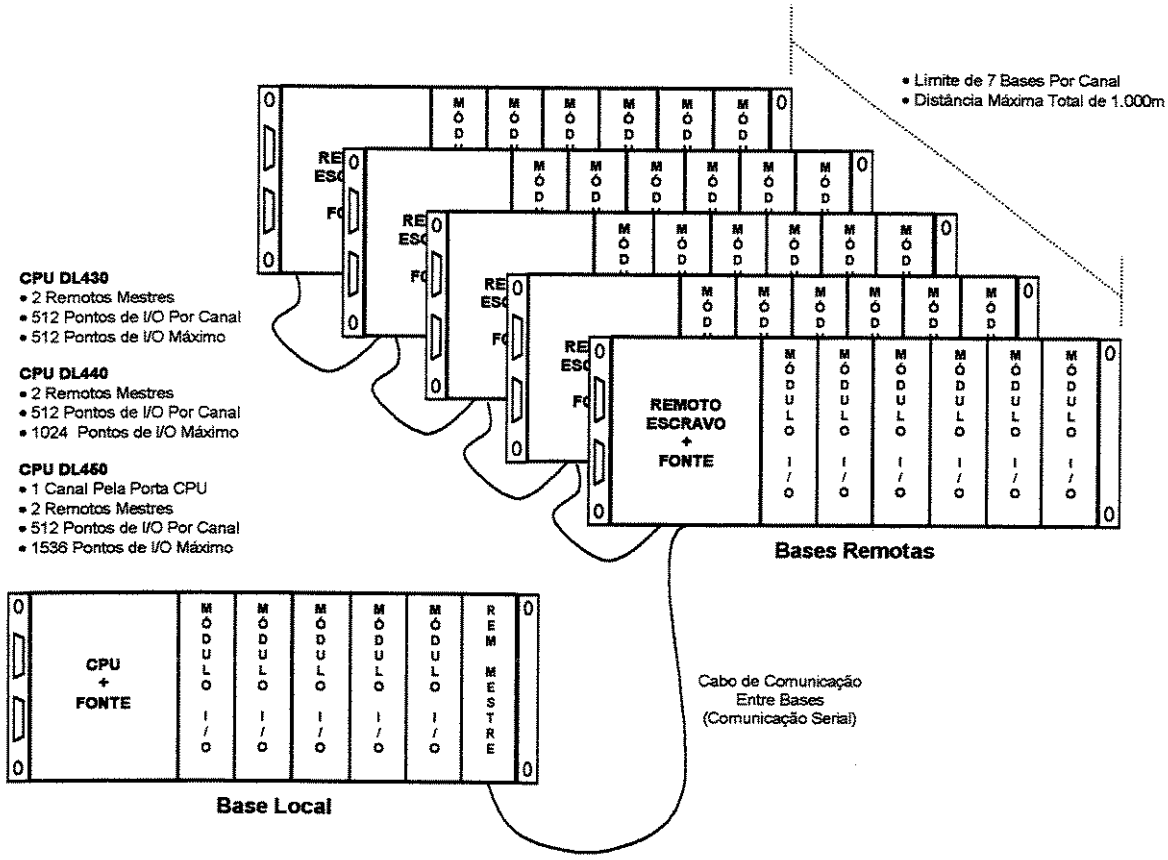
Expansão Remota

Esta configuração é utilizada quando os dispositivos de entrada e/ou saída estão localizados (instalados) distantes da Base Local, ou quando se necessita de Pontos de I/O em quantidade superior a que a Base Local e as Bases de Expansão Locais podem suportar.

Na(s) Base(s) de Expansão Remota(s) não há CPU, apenas a Fonte de Alimentação e um Módulo Especial de Comunicação – Módulo Remoto Escravo, que pode se apresentar como um único Módulo em conjunto com a fonte, conforme a Família de PLC. Na Base Local é instalado o Módulo Remoto Mestre, que proporciona um Canal de Comunicação Serial para acesso à(s) Base(s) de Expansão Remota(s).

As Famílias de PLCs que permitem esta configuração, especificam a quantidade limite de Módulos Remotos Mestres que podem ser instalados na Base Local. Algumas CPUs porporcionam um Canal para acesso à(s) Base(s) de Expansão Remota(s) através de porta de comunicação especial existente na própria CPU. Especificam, também, a quantidade limite de

Bases de Expansão, com distância máxima entre elas, e de Pontos de I/O possíveis para cada Canal de Comunicação. A figura 3.23 exemplifica esta situação.



Expansões Remotas – Família DL405, *PLC Direct by Koyo*

Figura 3.23 – Configuração do Sistema de I/O – *Expansão Remota*

A utilização de Bases de Expansão Remotas não restringe a utilização de Bases de Expansão Locais. Um mesmo sistema pode ser configurado com ambas opções de expansão.

A forma de identificação, o método de endereçamento e o tratamento dos Pontos de I/O Remotos dependem da Família de PLC utilizada. De maneira geral, os Pontos de I/O Remotos não são tratados como os Pontos de I/O Locais. As operações de leitura e escrita dos Pontos de I/O Remotos são feitas pela CPU através do Módulo Remoto Mestre, que se comunica com os Módulos Remotos Escravos. Não há sincronismo entre o acesso da CPU ao Módulo Remoto Mestre – que ocorre a cada *scan*, e o acesso deste aos Módulos Remotos Escravos – que depende da quantidade de Módulos e de Pontos de I/O instalados, e da taxa de transmissão (*baud rate*)

utilizada. Por este motivo, a atualização dos Pontos de I/O Remotos pode ser mais lenta que a atualização dos Pontos de I/O Locais (Bases Local e de Expansão Local).

3.4.2 Cálculo de Consumo de Potência

Embora o cálculo envolva apenas a variável corrente, utiliza-se "Cálculo de Consumo de Potência" pelo fato de estar relacionada a determinada tensão, normalmente de 5VDC (P = U . I).

Cada Módulo de I/O, utilizado na configuração de um PLC, consome determinada corrente do barramento da Base para poder operar, a qual é fornecida pela Fonte de Alimentação. Nos casos em que a CPU se apresenta como módulo independente (não acoplado à fonte), também esta consome tal corrente. Portanto, faz-se necessário, durante a configuração de um PLC, determinar qual a corrente consumida por todos os módulos que o compõem, e se a mesma não ultrapassa o limite de corrente que a Fonte de Alimentação pode fornecer. Durante este cálculo, é aconselhável verificar a necessidade de alimentação externa para os Módulos de I/O, determinando tensão e corrente utilizadas.

Por exemplo, como visto na figura 3.19, a Família DL205 da *PLC Direct by Koyo* proporciona quatro modelos de Base - com Fonte de Alimentação acoplada, para configuração do PLC com 3, 4, 6 ou 9 *slots*. As Bases de 3, 4 e 6 *slots* têm Fonte de Alimentação que fornece corrente de até 1.550mA ao barramento. Já a Base de 9 *slots* tem este valor fixado em 2.600mA. Conforme a configuração do PLC, pode ocorrer que uma Base com 6 *slots* seja adequada para alojar a CPU e os Módulos de I/O utilizados, mas a corrente consumida ultrapasse os 1.550mA fornecidos pela Fonte de Alimentação. Neste caso, é necessário optar pela Base de 9 *slots* por 'motivos elétricos'.

A seguir, são demonstrados 'Cálculos de Consumo de Potência' para duas configurações de PLCs utilizando a Família DL205 da *PLC Direct by Koyo*.

- **Configuração 1:** CPU DL240, 16 Pontos de Entrada 24VDC (*current sinking*) e 48 Pontos de Saída 24VDC (100mA/ponto, *current sinking*).

MÓDULO	CARACTERÍSTICAS	Corrente Consumida (5VDC)	Alimentação Externa (24VDC)
D2-240	CPU DL240	120mA	------
D2-16ND3-2	Módulo 16 Pontos de Entrada, 24VDC, *current sinking/sourcing*, 2 comuns isolados	100mA	------
D2-16TD1-2	Módulo 16 Pontos de Saída, 12-24VDC, *current sinking*, 100mA/ponto, 2 comuns interligados	200mA	80mA

Portanto, o PLC apresentará a seguinte configuração:

SLOT	MÓDULO	5 VDC (mA)	24VDC (mA) - Externo -
CPU	D2-240	120	------
0	D2-16ND3-2	100	------
1	D2-16TD1-2	200	80
2	D2-16TD1-2	200	80
3	D2-16TD1-2	200	80
	Consumo Total	820	240

Para atender à configuração apresentada, são necessários **5 *slots*** para instalação da CPU e de 4 Módulos de I/O, sendo consumidos **820mA** do barramento da Base (5VDC) e **240mA** da fonte externa (24VDC). Assim, conclui-se que a Base com 6 *slots* (D2-06B) é adequada a esta configuração, permanecendo um *slot* vazio. Porém, a saída auxiliar de 24VDC da Fonte de Alimentação da Base (200mA) não pode ser utilizada para alimentação dos Módulos de Saída (240mA), sendo necessário utilização de Fonte Externa (a mesma que alimentará os dispositivos de entrada/saída).

- **Configuração 2:** CPU DL240, 16 Pontos de Entrada 24VDC (*current sinking*) e 48 Pontos de Saída 110VAC (200mA/ponto).

MÓDULO	CARACTERÍSTICAS	Corrente Consumida (5VDC)	Alimentação Externa (24VDC)
D2-240	CPU DL240	120mA	------
D2-16ND3-2	Módulo 16 Pontos de Entrada, 24VDC, *current sinking/sourcing*, 2 comuns isolados	100mA	------
D2-12TA	Módulo 12 Pontos de Saída, 15-132VAC, 300mA/ponto, 2 comuns isolados	350mA	------

Portanto, o PLC apresentará a seguinte configuração:

SLOT	MODULO	5 VDC (mA)	24VDC (mA) - Externo -
CPU	D2-240	120	------
0	D2-16ND3-2	100	------
1	D2-12TA	350	------
2	D2-12TA	350	------
3	D2-12TA	350	------
4	D2-12TA	350	------
	Consumo Total	1.620	------

Para atender à configuração apresentada, são necessários **6** ***slots*** para instalação da CPU e de 5 Módulos de I/O, sendo consumidos **1.620mA** do barramento da Base (5VDC). Assim, conclui-se que a Base com 9 *slots* (D2-09B) é adequada a esta configuração, permanecendo três *slots* vazios, uma vez que a Fonte de Alimentação da Base com 6 *slots* (fisicamente adequada) tem o limite de corrente em 1.550mA.

O mesmo cuidado deve ser tomado durante a configuração de PLCs que apresentem a Fonte de Alimentação como módulo independente ou acoplada à CPU.

3.4.3 Configuração da Porta Serial

De acordo com a aplicação, pode ser necessário ou desejado, a conexão do PLC com Interfaces Homem-Máquina (IHMs), Computadores Pessoais (PCs) ou outros PLCs (Rede de PLCs), por exemplo. Nestes casos, faz-se necessário configurá-lo corretamente para atender às necesidades de comunicação exigidas em cada situação.

Normalmente, as CPUs contém pelo menos uma porta de comunicação serial que pode ser conectada aos dispositivos externos. A quantidade de portas varia de acordo com a CPU, podendo chegar a quatro. Porém, as características de configuração destas portas, tanto de hardware como de software, podem não ser compatíveis com as necessárias em determinada aplicação. Nesta situação, deve-se incluir na configuração do PLC um Módulo Especial de Comunicação, com as características desejadas, desde que a Família de PLC utilizada disponha do mesmo, ou então, optar por uma Família que atenda completamente às necessidades.

De maneira geral, as características que devem ser observadas durante a configuração do PLC, quanto à Porta Serial, são:

1. Hardware

- Padrão da Porta Serial. Normalmente, são encontrados:

  RS-232: padrão EIA (*Electronic Industries Association*) para transmissão de dados através de cabo 'par-trançado', em distâncias de até 15m. Define pinagem de conectores, níveis de sinais, impedância de carga, etc, para dispositivos de transmissão e recepção. É o padrão existente nas portas seriais dos PCs

  RS-422: padrão EIA para transmissão de dados com balanceamento de sinal (linhas de transmissão e recepção têm comuns independentes), proporcionando maior imunidade a ruídos, maior velocidade de transmissão e distâncias mais longas (até 1.200m). A comunicação é *full-duplex* (pode enviar e receber dados simultaneamente)

  RS-485: similar ao padrão RS-422. Os receptores têm proteções e capacidades maiores. A comunicação é *half-duplex* (pode apenas enviar ou receber dados em um mesmo instante)

Se a CPU proporcionar Porta Serial padrão RS-232 e for necessário o padrão RS-422 ou RS-485 - por motivos de compatibilidade com o dispositivo ao qual será conectada, ou pela distância ser superior a 15m, ou ainda para garantir maior imunidade a ruídos - utiliza-se um conversor RS232/422 ou RS232/485, sem necessidade de adicionar Módulo Especial de Comunicação, ou trocar a Família de PLC. O mesmo acontece se a situação for inversa, a CPU proporcionar Porta Serial padrão RS-422 / RS-485 e se desejar conexão com PC (RS-232).

Algumas CPUs utilizam padrão próprio de comunicação, sendo necessário a utilização de hardware dedicado fornecido pelo fabricante.

2. Software

- Protocolo de Comunicação. Determina a forma de transmissão dos dados (formato dos dados, temporização, sinais de controle utilizados, etc). Cada fabricante de PLC tem seu Protocolo de Comunicação próprio, normalmente chamado de 'Protocolo Proprietário', o qual é utilizado durante a programação do PLC. Alguns são 'abertos' (o usuário tem acesso

ao formato da transmissão de dados utilizada, podendo desenvolver seus próprios programas de comunicação) e outros são 'restritos' (o fabricante não fornece informações sobre o Potocolo Proprietário). No entanto, há CPUs que, além de suportarem o Protocolo Proprietário, suportam protocolos padrões (*Modbus*, *Profibus*, *Devicenet*, por exemplo), permitindo comunicação com dispositivos e softwares fornecidos por outros fabricantes, além de conexão em rede

- Taxa de Transmissão (*Baud Rate*). Determina a velocidade, expressa em bits por segundo (*bps – bits per second*), da transmissão dos dados

As características de software devem ser as mesmas, tanto na CPU do PLC como nos dispositivos a ela conectados. Porém, se a CPU não puder ser configurada para suportar o protocolo ou a taxa de transmissão desejados, deve-se optar pela utilização de Módulo Especial de Comunicação – desde que suporte as características necessárias, ou pela utilização de uma Família de PLCque atenda completamente às necessidades.

3.5 Ferramentas de Programação dos Controladores Lógicos Programáveis

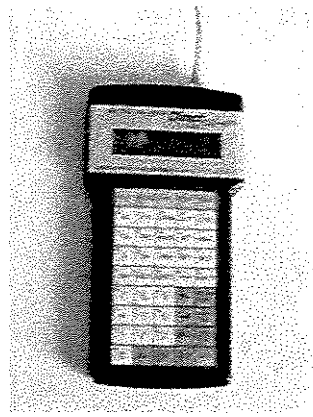
As principais Ferramentas de Programação atualmente encontradas, para as Famílias de PLCs disponíveis no mercado, são o Programador Manual (*Handheld Programmer*) e o Software de Programação (que deve ser instalado em microcomputador apropriado). Essas ferramentas permitem, além da programação do PLC, monitoramento de condições internas à CPU (diagnósticos e erros, por exemplo), verificação da execução do Programa de Aplicação e controle sobre modos de operação, entre outros recursos.

Cada fabricante, e em alguns casos cada Família de PLC, tem sua Ferramenta de Programação própria, que não pode ser usada para PLCs (ou CPUs) distintos.

3.5.1 Programador Manual (*Handheld Programmer*)

Esta é a ferramenta de menor custo e utilizada para pequenas alterações. Normalmente, possui um display de cristal líquido com duas linhas para apresentação das informações (endereço e dados do programa, condição dos Pontos de I/O e diagnósticos internos, por

exemplo) e um teclado de membrana para entrada dos dados. A figura 3.24 apresenta um exemplo de Programador Manual.



Programador Manual para algumas Famílias da *PLC Direct by Koyo*

Figura 3.24 – Ferramentas de Programação – *Programador Manual*

O Programador Manual não é indicado para o desenvolvimento de todos os Programas de Aplicação, pois permite edição/alteração através de mnemônicos (Linguagem de Lista de Instruções) apenas. Porém, é bastante útil como Ferramenta de Manutenção para campo ('chão de fábrica'), proporcionando visualização, monitoramento e alteração de parâmetros e do Programa de Aplicação rapidamente, com a vantagem de ser portátil e resistente ao ambiente industrial. É conectado à CPU do PLC através de cabo apropriado, pelo qual recebe a tensão de alimentação necessária à sua operação.

Algumas Famílias de Micro PLCs permitem o desenvolvimento de programas apenas através desta Ferramenta de Programação. Conforme o fabricante, o *backup* (cópia de segurança) do Programa de Aplicação desenvolvido pode ser armazenado em cartões de memória tipo PCMCIA, ou em memórias tipo EEPROM, ambos instalados no próprio Programador Manual.

3.5.2 Software de Programação

É a ferramenta mais poderosa disponível no mercado. Conforme o PLC, o Software de Programação opera em ambiente DOS® ou Windows®, sendo este o mais comum atualmente. Além de proporcionar edição/alteração do Programa de Aplicação em ambiente gráfico

(Linguagem Ladder, por exemplo) – mesmo para as versões DOS®, permite visualização e controle total do sistema, documentação e impressão da aplicação desenvolvida, forma de armazenamento de *backup* conforme o PC utilizado (disquete, HD, CD), e recursos avançados para depuração e manutenção. O microcomputador usado deve atender às configurações de hardware (processador, quantidade de memória RAM, espaço livre em HD, portas seriais) e software (Sistema Operacional) indicadas pelo fabricante do PLC.

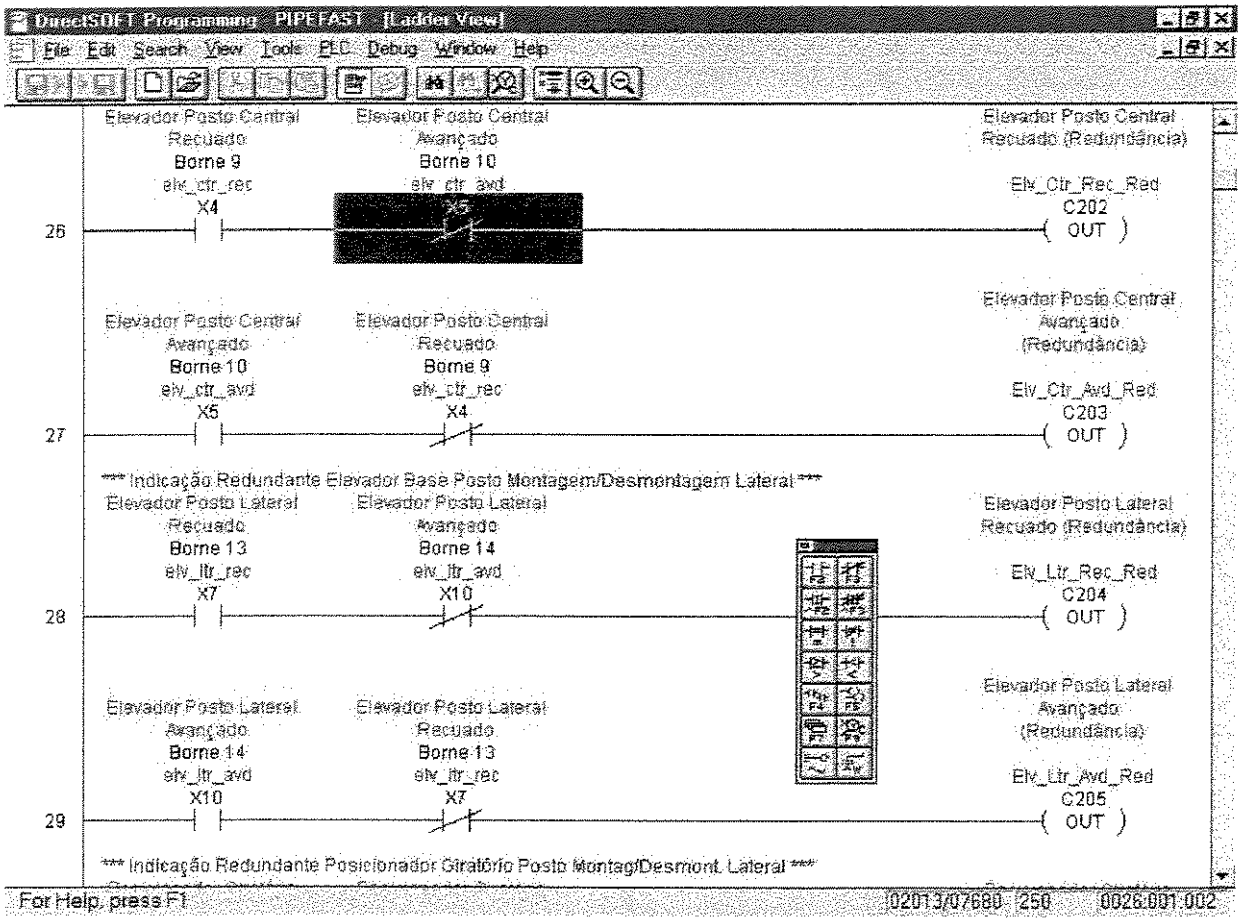
Conforme o Software de Programação, são disponíveis dois modos de operação:

- *OFF-LINE* (Sem Conexão): permite o desenvolvimeto do Programa de Aplicação (edição, documentação, impressão) e configuração de parâmetros sem necessidade de conexão com a CPU do PLC
- *ONLINE* (Conectado): os recursos são disponíveis a partir da conexão com a CPU do PLC. Alguns Softwares de Programação permitem operação apenas neste modo, ou seja, todo o desenvolvimento deve ser realizado com o microcomputador conectado ao PLC

A comunicação entre o microcomputador e a CPU do PLC é feita através de cabo apropriado, pela porta serial (RS-232) na maioria dos casos. Porém, algumas CPUs utilizam o padrão RS-422 e necessitam de conversor RS-232/RS-422 para conexão. Há ainda, aquelas que utilizam padrão próprio e necessitam de interface dedicada instalada no PC.

Os recursos e facilidades que o Software de Programação oferece, variam conforme o fabricante. Por exemplo, o Software de Programação *Direct*SOFT, da *PLC Direct by Koyo*, opera em ambiente Windows® (a partir da versão 3.1), proporciona nos modos *Off-Line* e *Online* poderosos recursos de edição, documentação e depuração/manutenção. Por operar em ambiente Windows®, permite a visualização de várias janelas simultaneamente, permitindo que dois ou mais Programas de Aplicação sejam criados/editados ao mesmo tempo, e que recursos de 'Marcar, Recortar, Colar' sejam utilizados entre eles. A comunicação com a CPU do PLC pode ser feita através de porta serial padrão RS-232, ou através de Modem, com busca e configuração automáticas em ambos os casos. Para comunicação via Modem – que permite a manutenção, alteração e atualização de aplicações à distância, são necessários dois Modems, um Modem instalado no PC (interno ou externo) e outro instalado no PLC (externo), configurados adequadamente.

A figura 3.25 apresenta a janela de edição de um Softtware de Programção em ambiente Windows®.



Software de Programação *Direct*SOFT – Para todas CPUs *PLC Direct by Koyo*

Figura 3.25 – Ferramentas de Programação – *Software de Programação*

3.6 Linguagens de Programação dos Controladores Lógicos Programáveis

A Linguagem Ladder foi a primeira linguagem destinada à programação de PLCs, sendo ainda hoje a mais utilizada. Trata-se de uma linguagem gráfica baseada em símbolos semelhantes aos encontrados nos esquemas elétricos (contatos e bobinas), tendo sido motivo determinante para aceitação do PLC por técnicos e engenheiros acostumados com os sistemas de controle a relés.

O nome Ladder deve-se à representação da linguagem se parecer com uma escada (*ladder*), na qual duas linhas verticais paralelas são interligadas pela lógica de controle (*rung*), formando os degraus da escada.

Atualmente, os PLCs apresentam instruções sofisticadas. Além de simples contatos e bobinas, dispõem de contatos para detecção de borda de subida/descida (*one shot* – 'disparo'), contatos de comparação, temporizadores, contadores, blocos de processamento (operações lógicas e aritméticas, manipulação de dados), controle total do fluxo de execução do programa (*loops For/Next*, *Goto*, *Stop*, sub-rotinas) e interrupções (por hardware e software), por exemplo. A figura 3.26 apresenta um exemplo simples de programa em Linguagem Ladder.

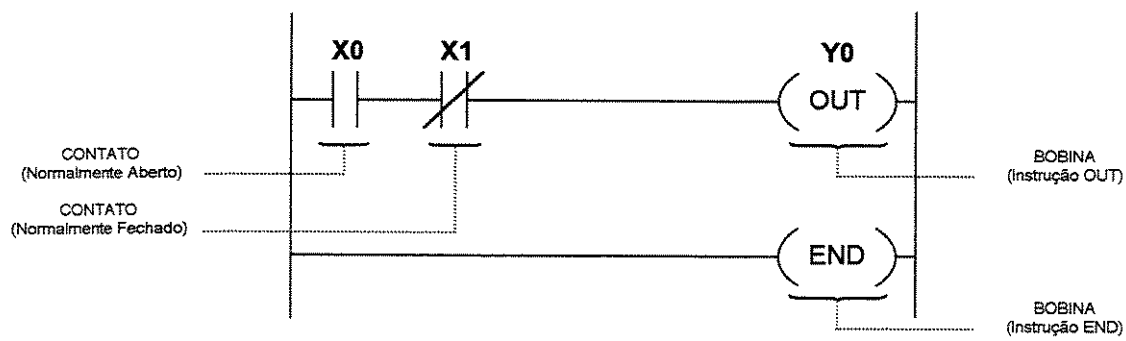


Figura 3.26 – Exemplo de Programação em Linguagem Ladder

Existem outras Linguagens de Programação para PLCs. A Norma IEC 1131-3 (1992), define cinco linguagens como padrões para programação de PLCs, entre elas a Linguagem Ladder. As demais são:

- **Diagrama Funcional Seqüencial (*SFC – Sequential Function Chart*)** - é uma linguagem gráfica que proporciona a representação das seqüências do programa através de um diagrama. Suporta seleções de seqüências alternativas e seqüências paralelas. Os elementos básicos são as etapas e as transições. As etapas consistem de uma 'porção' do programa que é executada até que a condição especificada na transição seja alcançada. Como a maioria das aplicações industriais executam uma seqüência de etapas, o *SFC* é uma forma lógica de especificar e programar PLC em alto nível, permitindo uma 'tradução' direta da descrição realizada do Sistema Automatizado. A figura 3.27 apresenta um exemplo simples de programa em Linguagem de Diagrama Funcional Seqüencial

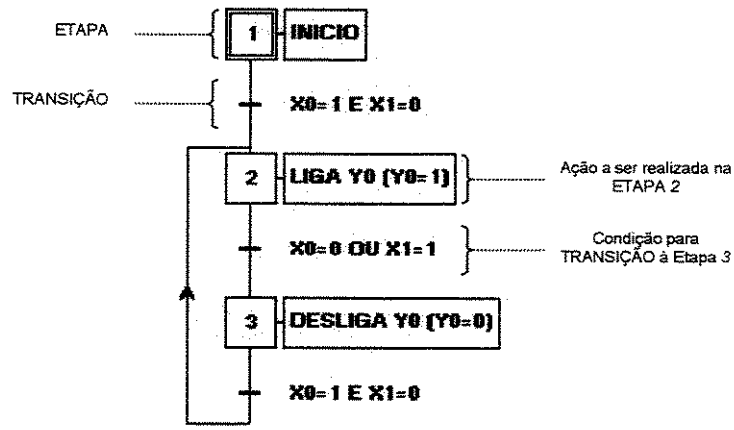


Figura 3.27 – Exemplo de Programação em Linguagem de Diagrama Funcional Seqüencial

- **Lista de Instruções (*IL – Instruction List*)** - é uma linguagem de baixo nível, similar à linguagem assembly. Nesta Linguagem, é permitida apenas uma operação por linha, como o armazenamento de um Valor em uma determinada Variável, por exemplo. Sua utilização é viável em aplicações menores, ou para otimização de partes de uma aplicação mais complexa. A figura 3.28 apresenta um exemplo simples de programa em Linguagem de Lista de Instruções

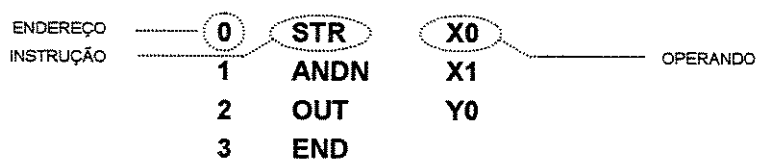


Figura 3.28 – Exemplo de Programação em Linguagem de Lista de Instruções

- **Texto Estruturado (*ST – Structured Text*)** - é uma linguagem de alto nível, estruturada em blocos, que tem uma sintaxe semelhante à do Pascal. Esta linguagem pode ser usada para expressar declarações complexas involvendo variáveis que representem uma ampla faixa de dados de diferentes tipos, incluindo valores analógicos e digitais. Há também, tipos de dados específicos para gerenciamento de horas, datas e durações de tempo, utilizados em aplicações de processamento de produtos, por exemplo. A linguagem suporta: interação de *loops*, como por exemplo REPEAT UNTIL (repita até que); execução condicional através

de construções usando IF-THEN-ELSE (se-então-senão); e funções tais como SQRT() (raiz quadrada de) e SIN() (seno de)

- **Diagrama de Blocos de Função (*FBD – Function Block Diagram*)** - é uma linguagem gráfica que permite aos elementos do programa, representados como blocos, serem conectados entre si de forma semelhante a um diagrama de circuito elétrico. Esta linguagem é apropriada para aplicações que envolvam fluxo de informação, ou dado, entre os componetes de controle

Para estar em conformidade com a norma IEC 1131-3, o PLC deve proporcionar a programação através destas cinco linguagens e permitir que o Programa de Aplicação seja desenvolvido utilizando todas elas, ou seja, ter partes do programa em Linguagem Ladder e partes em Linguagem de Lista de Instruções, podendo ter as cinco linguagens utilizadas conforme a escolha do programador. Esta norma tem gerado conflito de opiniões desde a publicação da primeira revisão em 1993, motivo pelo qual não se tornou ainda um padrão para os fabricantes de PLC.

Além das Linguagens de Programação definidas na norma, são encontrados no mercado PLCs que proporcionam programação através de outras linguagens -'C'e BASIC, entre elas. Por exemplo, as CPUs *PLC Direct by Koyo* permitem programação através de Linguagem Ladder (Software *Direct*SOFT), Lista de Instruções (Programador Manual, e apenas visualização através do *Direct*SOFT) e criação de Diagrama Funcional Seqüencial através da Linguagem Ladder. Portanto, estas CPUs ainda não atendem à norma IEC 1131-3.

3.7 Sistema de Operação do Controlador Lógico Programável

O gerenciamento de todo o sistema pela CPU do PLC é feito de forma específica, de acordo com o *firmware* de cada fabricante. O conhecimento e entendimento do mesmo interfere diretamente no desenvolvimento de Programas de Aplicação eficazes.

Com pequenas variações, pode-se considerar o fluxograma apresentado na figura 3.29 como válido para a maioria dos PLCs encontrados no mercado. Alguns poderão ter segmentos além dos apresentados, ou não ter todos estes (a CPU que não disponha de relógio interno não

terá o segmento de 'Atualização de Relógio/Calendário', por exemplo), ou ainda apresentar pequenas alterações quanto à seqüência dos mesmos.

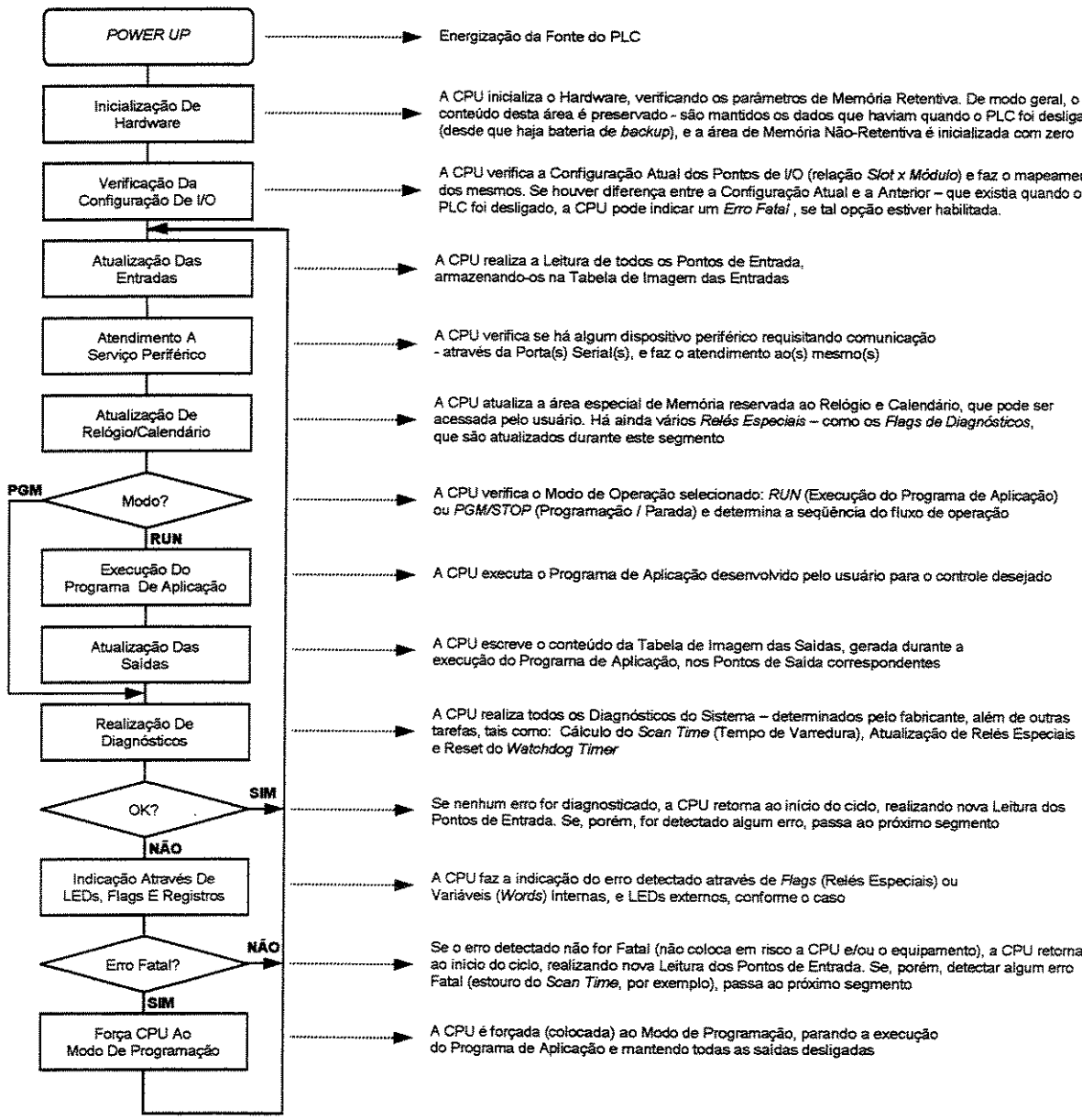


Figura 3.29 – Fluxograma Típico do Sistema de Operação de um PLC

De maneira geral, as CPUs apresentam dois Modos de Operação:

- *Programação* (*Program* ou *PGM*): neste modo, a CPU não executa o Programa de Aplicação e não atualiza os Pontos de Saída. A função principal deste modo é permitir a transferência e/ou alteração do Programa de Aplicação, através da Ferramenta de

Programação utilizada. Permite, também, a configuração de parâmetros da CPU (*Set Up*), como por exemplo a determinação da área de Memória Retentiva

- *Execução* (*RUN*): neste modo, a CPU executa o Programa de Aplicação desenvolvido pelo usuário para realização do controle desejado, atualizando os Pontos de Saída conforme a lógica programada

Os segmentos "Inicialização de Hardware" e "Verificação da Configuração de I/O" são executados apenas uma vez, após a energização ("Power-Up") da Fonte do PLC. Os demais segmentos, a partir da "Atualização das Entradas", formam o Ciclo de Execução do PLC (*scan* – varredura). Alguns segmentos são executados conforme o Modo de Operação da CPU, ou pela ocorrência de situações específicas.

A alteração entre os Modos de Operação pode ser feita através de chave seletora na própria CPU, ou através de Ferramenta de Programação (Programador Manual ou Software de Programação). Algumas CPUs têm, além das opções de *PGM* e *RUN*, outras posições para a chave, por exemplo: *TERM* (indica que o Modo de Operação será determinado por um dispositivo externo - '*Terminal*', ou seja, pela Ferramenta de Programação) e *STOP* (força a CPU ao Modo de Parada, porém não permite a alteração do Programa de Aplicação).

A seguir, são apresentados com mais detalhes os principais segmentos do Fluxograma do Sistema de Operação da CPU, existentes em todos os PLCs.

3.7.1 Atualização das Entradas - *Leitura das Entradas*

A CPU realiza a Leitura de todos os Pontos de Entrada e armazena-os na Tabela de Imagem das Entradas. Cada Ponto de Entrada corresponde a uma posição de memória específica (um *bit* de uma determinada *word*), conforme ilustrado na figura 3.30.

A Tabela de Imagem das Entradas é acessada pela CPU durante a execução do Programa de Aplicação. Como visto na figura 3.29 (Fluxograma do Sistema de Operação da CPU), após a execução deste segmento em um determinado *scan*, a Leitura das Entradas será realizada apenas no *scan* seguinte, ou seja, se o *status* (condição) de um determinado Ponto de Entrada mudar após

a Leitura das Entradas, ele só terá influência na execução do Programa de Aplicação no *scan* seguinte, quando será percebida tal alteração.

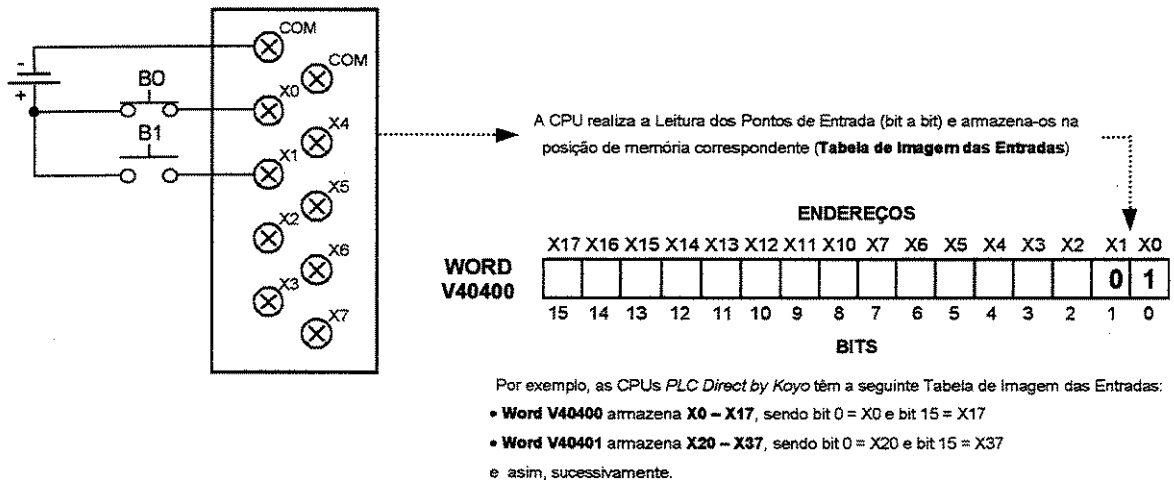


Figura 3.30 – Atualização das Entradas

Se uma determinada aplicação não puder 'esperar' este tempo (normalmente, da ordem de milisegundos) para reconhecimento da alteração dos Pontos de Entrada, utiliza-se Instruções Imediatas para construção da Lógica de Controle no Programa de Aplicação. Estas instruções acessam diretamente os Pontos de Entrada no momento em que são executadas. Há também as Instruções Imediatas de Saída que, ao serem executadas, atualizam os Pontos de Saída e a Tabela de Imagem das Saídas simultaneamente. A utilização de Instruções Imediatas aumenta o *Scan Time* da CPU, pois além das operações de Atualização das Entradas e Atualização das Saídas, os Módulos de I/O são acessados a cada execução de uma Instrução Imediata.

3.7.2 Execução do Programa de Aplicação

Neste segmento, a CPU executa as instruções do Programa de Aplicação, que definem a relação entre a condição das Entradas e a atuação das Saídas, ou seja, definem a Lógica de Controle a ser realizada.

A CPU inicia a execução do Programa de Aplicação a partir do primeiro *rung* (Lógica de Controle da Linguagem Ladder), executando-o da esquerda para a direita, e de cima para baixo, *rung* a *rung*, até encontrar a instrução **END** (FIM). Constrói, assim, uma nova Tabela de Imagem

das Saídas, gerada a partir da lógica executada. A figura 3.31 apresenta a execução do Programa de Aplicação, conforme descrito.

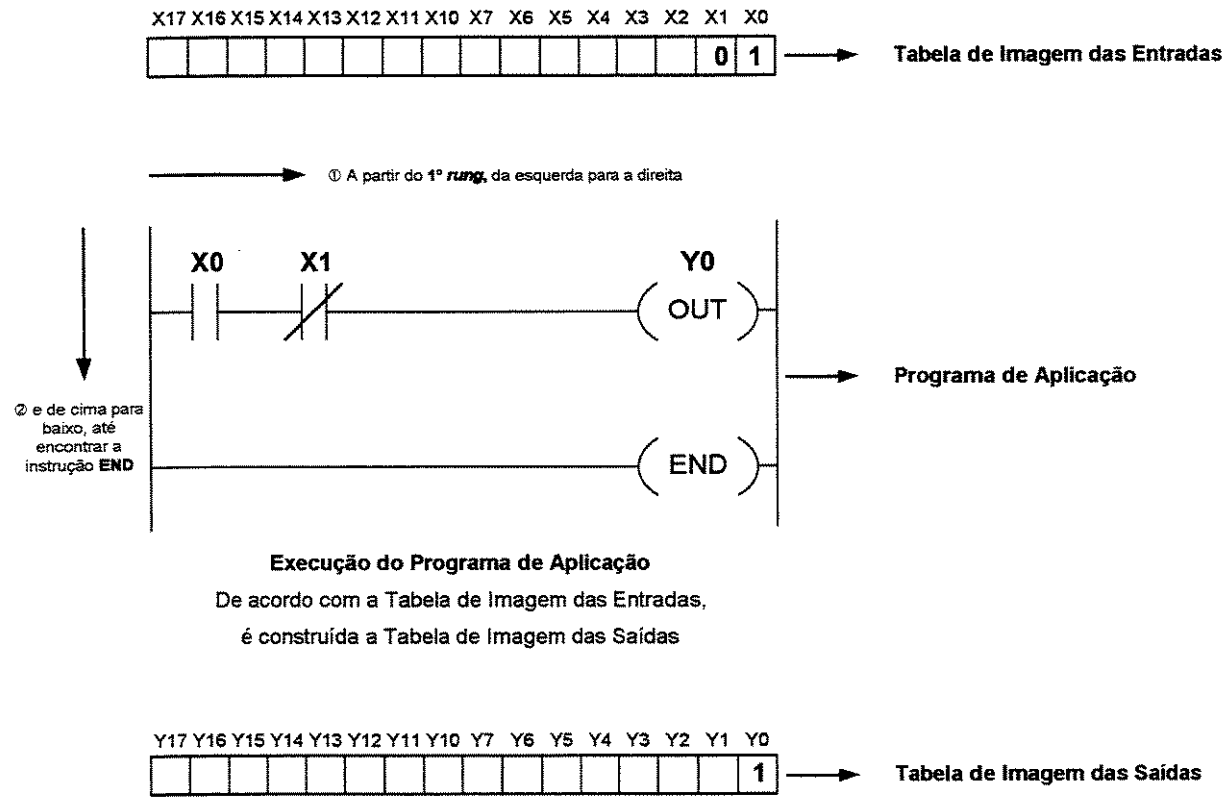


Figura 3.31 – Execução do Programa de Aplicação

3.7.3 Atualização das Saídas - *Escrita das Saídas*

Após a execução do Programa de Aplicação, o conteúdo da Tabela de Imagem das Saídas, construída de acordo com a lógica executada, é enviado aos Pontos de Saída correspondentes, conforme apresentado na figura 3.32.

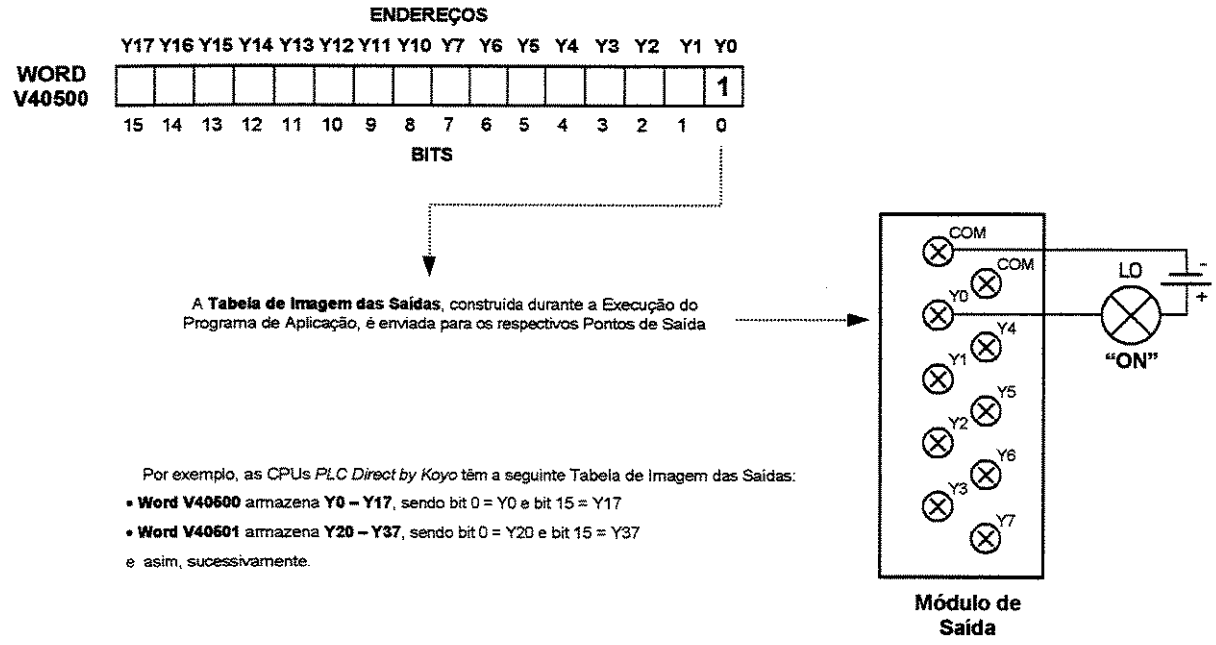


Figura 3.32 – Atualização das Saídas

3.7.4 Realização de Diagnósticos

Neste segmento, a CPU realiza todos os diagnósticos do sistema, além de calcular o *Scan Time* (Tempo de Varredura), atualizar os relés especiais e reinicializar o *Watchdog Timer* (Cão-de-Guarda do Relógio).

Entre os diagnósticos, os mais importantes realizados são o cálculo do *Scan Time* e o controle do *Watchdog Timer*. O *Scan Time* compreende o tempo consumido pela CPU para realizar todas as tarefas em cada *scan*, desde o início (Atualização das Entradas) até o término do ciclo (Atualização das Saídas). O *Watchdog Timer* armazena o tempo máximo permitido para execução de cada *scan* (normalmente definido pelo usuário). Se, em determinado *scan*, esse tempo for excedido (Erro Fatal), a CPU é forçada ao Modo de Programação e todas as saídas são desligadas. Caso contrário, o valor do *Scan Time* é armazenado em uma variável apropriada (para realização de estatísticas: *Scan Time* máximo e mínimo, por exemplo) e juntamente com o *Watchdog Timer* é reinicializado, sendo controlados a cada *scan*.

Todos os erros diagnosticados, Fatais ou Não-Fatais, são indicados por *flags* (bits internos à CPU, que podem ser usados no Programa de Aplicação) ou por LEDs externos (normalmente

localizados na parte frontal da CPU e dos Módulos de I/O), conforme o caso. Algumas CPUs dispõem, também, de uma variável destinada ao armazenamento do Código de Erro ocorrido no último *scan*.

Neste capítulo foi apresentada a arquitetura básica de um PLC, com descrição dos blocos que o compõem. Foram apresentadas as possíveis configurações utilizadas em implementações de Sistemas Automatizados, as ferramentas e as linguagens de programação, sendo finalizado com a descrição do sistema de operação interno.

No próximo capítulo serão apresentados os conceitos para programção de PLCs através da Linguagem Ladder, com apresentação das instruções básicas e da programação por estágios.

Capítulo 4

Programação de PLCs – *Linguagem Ladder*

Este capítulo apresenta os conceitos necessários para o desenvolvimento de Programas de Aplicação através da Linguagem Ladder. São apresentados os tipos de dados normalmente disponíveis nas CPUs, as instruções básicas e a programação por estágios (implementação do *SFC*). São utilizadas as seguintes referências bibliográficas: Celati (1995), PLC Direct (1997 e 1998) e Stenerson (1999).

Apesar de ter sido a primeira linguagem destinada à programação de PLCs, a Linguagem Ladder é ainda a mais utilizada, estando presente praticamente em todos os PLCs disponíveis no mercado. Por ser uma linguagem gráfica, baseada em símbolos semelhantes aos encontrados nos esquemas elétricos (contatos e bobinas), as possíveis diferenças existentes entre os fabricantes de PLCs, quanto à representação das instruções, são facilmente assimiladas pelos usuários, como exemplificado na figura 4.1.

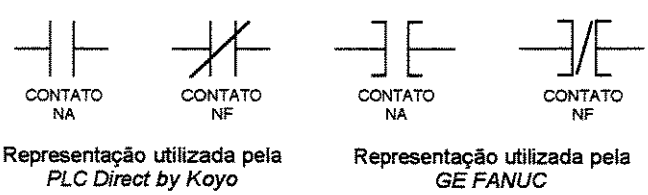


Figura 4.1 – Exemplo de Instruções em Linguagem Ladder

O nome Ladder deve-se à representação da linguagem se parecer com uma escada (*ladder*), na qual duas linhas verticais paralelas são interligadas pela Lógica de Controle, formando os degraus (*rung*) da escada. Portanto, a cada Lógica de Controle existente no Programa de Aplicação dá-se o nome de *rung*, o qual é composto por Colunas e Linhas, conforme apresentado na figura 4.2. A quantidade de Colunas e Linhas, ou Elementos e Associações, que cada *rung* pode conter é determinada pelo fabricante do PLC, podendo variar conforme a CPU utilizada. Em geral, este limite não representa uma preocupação ao usuário durante o desenvolvimento do Programa de Aplicação, pois atualmente os Softwares de Programação indicam se a quantidade estabelecida pelo fabricante foi ultrapassada, através de erro durante a compilação do Programa de Aplicação.

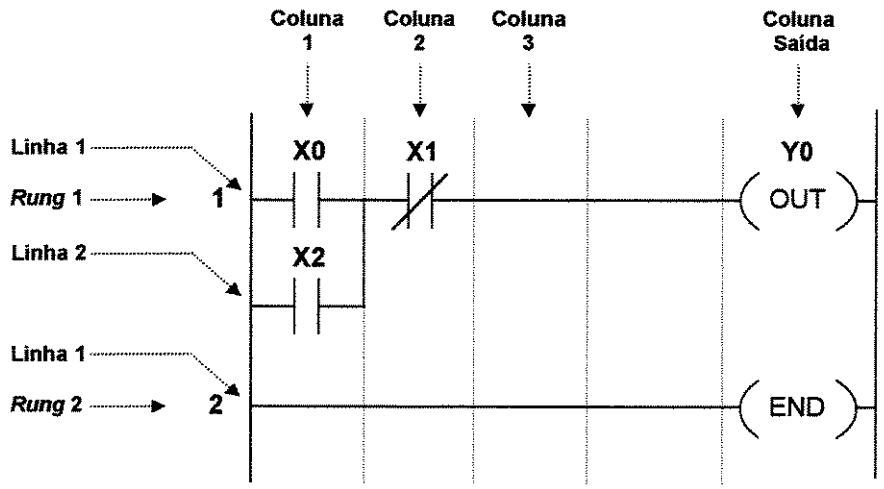


Figura 4.2 – Componentes da Programação em Linguagem Ladder

Cada Elemento (contato ou bobina, por exemplo) da Lógica de Controle representa uma Instrução da Linguagem Ladder, sendo alocada em um endereço específico e consumindo uma quantidade determinada de memória (*word*) disponível para armazenamento do Programa de Aplicação, conforme a CPU utilizada. Um mesmo símbolo gráfico da Linguagem Ladder (Contato Normalmente Aberto, por exemplo) pode representar Instruções diferentes, dependendo da posição ocupada na Lógica de Controle.

A figura 4.3 apresenta a equivalência entre o Programa de Aplicação em Linguagem Ladder e o mesmo em Lista de Instruções. Como pode ser visto, cada Instrução utilizada na Linguagem Ladder ocupou apenas um endereço de memória, o que é verificado pelo incremento simples de endereço em Linguagem de Lista de Instruções. Porém, há instruções que ocupam mais de um endereço de memória, conforme a CPU utilizada.

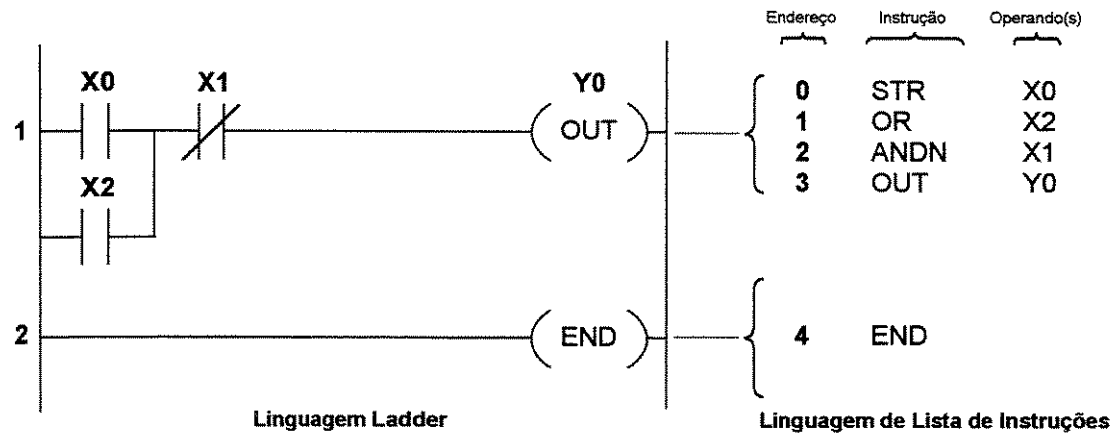


Figura 4.3 – Programa de Aplicação em Linguagem Ladder e em Lista de Instruções

A relação entre o símbolo gráfico da Linguagem Ladder e a Instrução a ser executada, pode ser verificada nos Endereços 0 e 1 da Linguagem de Lista de Instruções. Neste caso, a representação em Linguagem Ladder para os Elementos X0 e X2 são Contatos Normalmente Abertos idênticos. Porém, o posicionamento de cada um na Lógica de Controle determina Instruções diferentes ou seja, o Contato Normalmente Aberto de X0, por iniciar o *rung*, determina a Instrução '*Store*' (STR X0) e o Contato Normalmente Aberto de X2 (com representação gráfica idêntica), por estar em paralelo com X0, determina a Instrução '*Or*' (OR X2). Esta característica da Linguagem Ladder normalmente facilita o desenvolvimento do Programa de Aplicação, uma vez que o usuário precisa certificar-se apenas se associação desejada é aceita pela CPU utilizada, não se prendendo à Instrução propriamente dita.

4.1 Conceitos Básicos da Programação em Linguagem Ladder

Os conceitos apresentados a seguir são necessários para o correto desenvolvimento de Programas de Aplicação em Linguagem Ladder. Os mesmos são aplicados a todos os PLCs, independente de fabricante e de recursos disponíveis na CPU utilizada.

4.1.1 Instrução END

Todo programa em Linguagem Ladder deve ter uma Instrução END, indicando o final do mesmo. Trata-se de uma bobina, conforme apresentado na figura 4.4, e é classificada como Instrução de Controle do Programa.

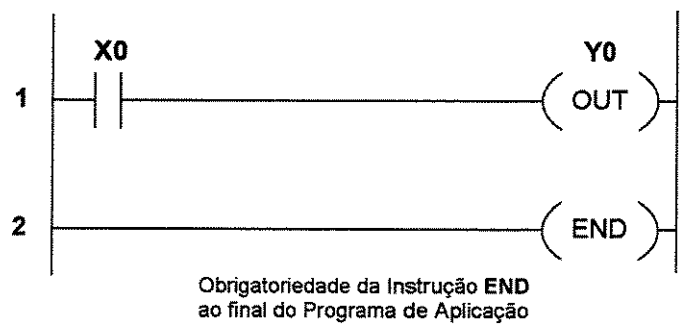


Figura 4.4 – Instrução END (Incondicional)

É uma Instrução incondicional, não admitindo qualquer tipo de Elemento em sua Lógica de Controle. Qualquer Instrução localizada após a Instrução END não será executada pelo Programa de Aplicação, com exceção às Instruções de Interrupção, Sub-Rotinas e Controles Específicos (Mensagens, por exemplo). A não existência da Instrução END no Programa de Aplicação gera um 'Erro Fatal', não permitindo à CPU entrar em Modo de Execução (*RUN*).

4.1.2 Corrente Lógica Fictícia

Para que uma bobina (ou outro Elemento de Saída - temporizador, contador ou bloco de função, por exemplo), seja acionada (executado) faz-se necessário 'energizá-la logicamente'. Assim, utiliza-se o conceito de Corrente Lógica Fictícia, ou seja, supondo que entre as linhas verticais (colunas) que sustentam a toda Lógica de Controle haja uma diferença de potencial (a coluna da esquerda com potencial positivo e a coluna da direita com potencial negativo, por

exemplo), haverá a circulação de corrente da esquerda para a direita se Lógica de Controle der condições para tal. A este efeito dá-se o nome de Corrente Lógica Fictícia.

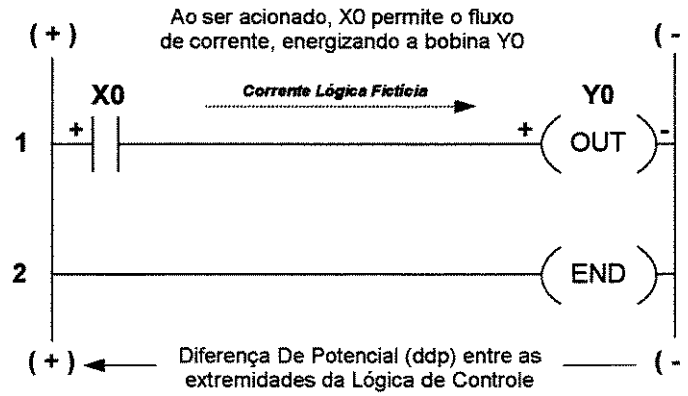


Figura 4.5 – Corrente Lógica Fictícia

Na figura 4.5 observa-se que se o Contato de X0 (Normalmente Aberto) estiver acionado – condição para que haja o fluxo de corrente entre as extremidades da Lógica de Controle, a bobina de Y0 será energizada, atuando tal Saída (através da instrução OUT). Caso contrário, a bobina de Y0 não é energizada (não estando submetida a uma 'ddp lógica'), mantendo a Saída desligada.

O sentido da Corrente Lógica Fictícia é sempre, e tão somente, da esquerda para a direita, não existindo a possibilidade de fluxo em sentido contrário.

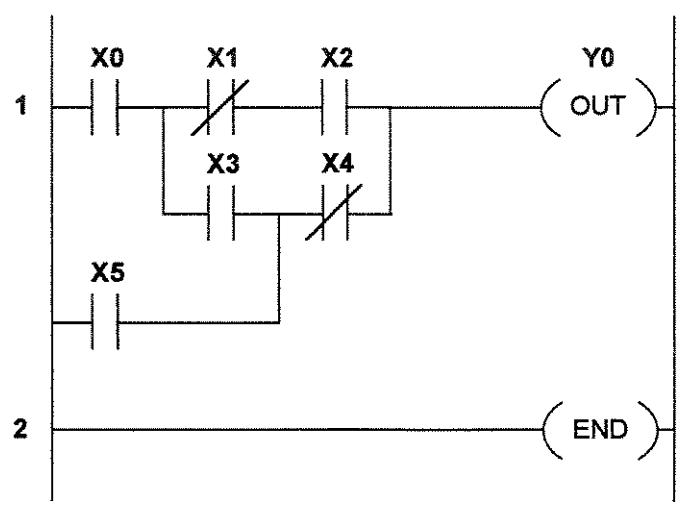


Figura 4.6 – Sentido da Corrente Lógica Fictícia (da Esquerda para a Direita)

No exemplo apresentado na figura 4.6, a bobina de Y0 pode ser acionada apenas através de X0-X1-X2, ou X0-X3-X4, ou X5-X4, devidamente acionados. Porém, nunca através de X5-X3-X1-X2, mesmo que devidamente acionados.

4.1.3 Implementação da Lógica de Controle

A Linguagem Ladder permite o acionamento de vários Elementos de Saída (bobinas, temporizadores, contadores, etc) simultaneamente, através da mesma Lógica de Controle, sem necessidade de construção de *rungs* similares.

A Lógica de Controle implementada na figura 4.7 determina que ao ser atuada a Entrada X0, as Saídas Y0, Y1 e Y2 serão acionadas simultanemante.

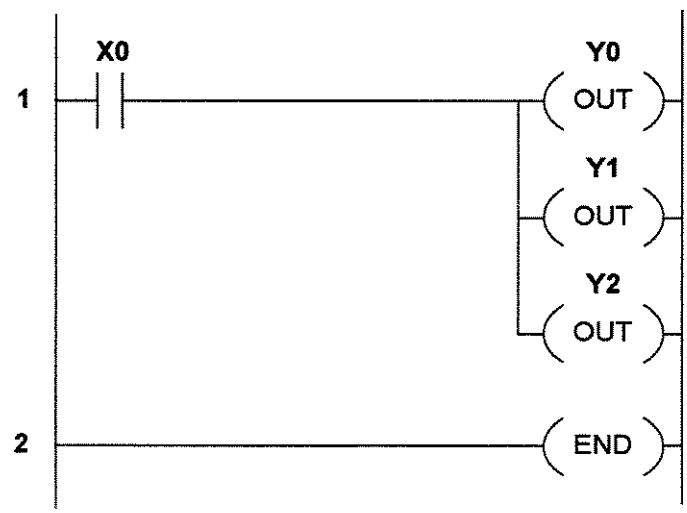


Figura 4.7 – Acionamento Simultâneo de Saídas Através da Mesma Lógica de Controle

É possível, também, implementar a Lógica de Controle utilizando derivações nas Linhas de um determinado *rung*. A Lógica de Controle apresentada na figura 4.8 determina que a Saída Y0 é acionada a partir da condição da Entrada X0 apenas (**Y0 = X0**), a Saída Y1 é acionada a partir da condição das Entradas X0 e X1 (**Y1 = X0 . X1**), e a Saída Y2 é acionada a partir da condição das Entradas X0, X1 e X2 (**Y2 = X0 . X1 . X2**). Com este recurso, evita-se a implementação de três Lógicas de Controle, em três *rungs* distintos.

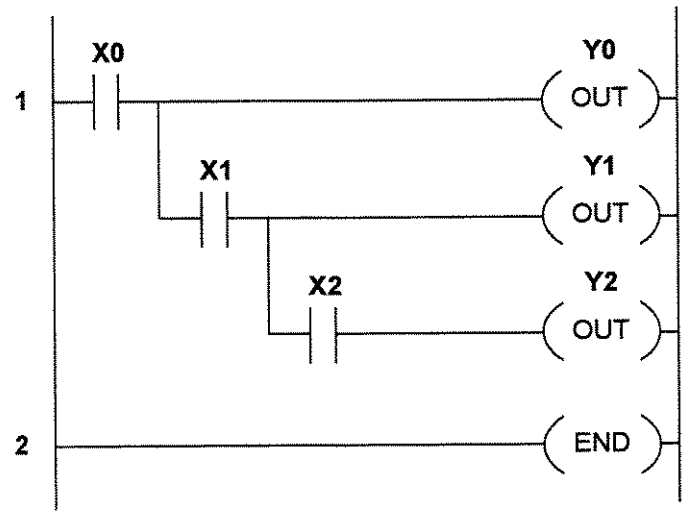


Figura 4.8 – Recurso para Implementação da Lógica de Controle

Algumas CPUs podem apresentar restrições quanto à utilização desta forma de implementação da Lógica de Controle. Por exemplo, as CPUs *PLC Direct by Koyo* não permitem que após a derivação, em qualquer Linha da Lógica de Controle, haja outro Elemento além da(s) Saída(s) controlada(s). Se isto ocorrer, haverá indicação de 'Erro' após a compilação do Programa de Aplicação. A figura 4.9 ilustra tal situação, na qual a Saída Y0 é acionada a partir da condição da Entrada X0 apenas (**Y0 = X0**), a Saída Y1 é acionada a partir da condição das Entradas X0 e X1 (**Y1 = X0 . X1**), e a Saída Y2 é acionada a partir da condição das Entradas X0 e X2 (**Y2 = X0 . X2**).

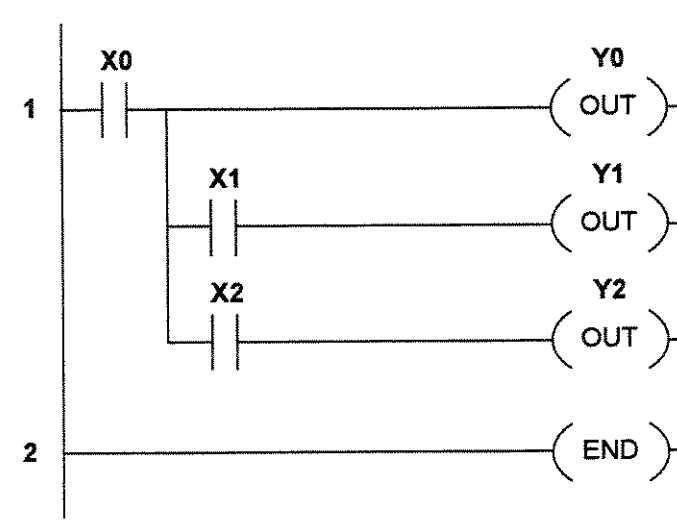


Figura 4.9 – Restrição, Encontrada em Algumas CPUs, quanto à Implementação da Lógica de Controle

Uma forma de implementar esta mesma Lógica de Controle, nas CPUs que apresentem tal restrição, é ilustrada na figura 4.10.

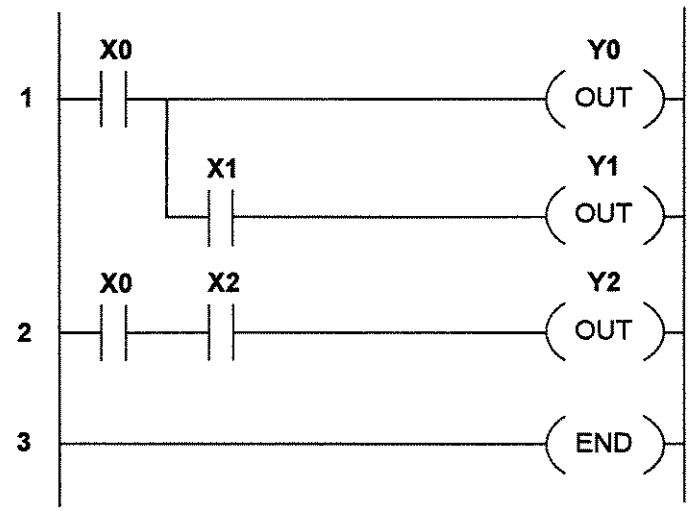


Figura 4.10 – Lógica de Controle sem Restrições para Implementação

4.1.4 Relação 'Dispositivos de Entrada' x 'Lógica de Controle'

A relação entre a condição do dispositivo de entrada (acionado ou não) e o Elemento utilizado na Lógica de Controle (Contato Normalmente Aberto ou Normalmente Fechado), pode causar certa confusão inicial ao usuário de PLC.

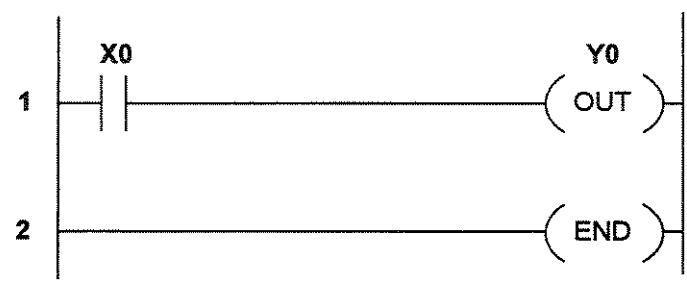


Figura 4.11 – Exemplo de Lógica de Controle Simples

Normalmente, faz-se a associação direta entre o Elemento utilizado na Lógica de Controle e a condição do dispositivo de entrada, o que gera tal confusão. Ou seja, ao se deparar com um programa que tenha Lógica de Controle semelhante à apresentada na figura 4.11, acredita-se inicialmente que a Saída Y0 estará acionada quando a Entrada X0 estiver aberta, tal qual indicado

na Linguagem Ladder. A verdade é exatamente oposta a esta idéia, ou seja, a Saída Y0 só estará acionada quando a Entrada X0 estiver fechada.

A relação existente entre a condição dos dispositivos de entrada e o Elemento utilizado na Lógica de Controle pode ser definida da seguinte maneira:

"Se o dispositivo de entrada estiver fechado (Ponto de Entrada / Tabela de Imagem das Entradas = 1), o Elemento utilizado na Lógica de Controle é atuado, ou seja o Contato Normalmente Aberto torna-se fechado (dando condição ao fluxo da Corrente Lógica Fictícia) e o Contato Normalmente Fechado torna-se aberto (impedindo o fluxo de tal corrente). Caso contrário, se o dispositivo de entrada estiver aberto (Ponto de Entrada / Tabela de Imagem das Entradas = 0), o Elemento utilizado na Lógica de Controle mantém seu estado natural (ou de repouso), sendo que o Contato Normalmente Aberto permanece aberto (impedindo o fluxo da Corrente Lógica Fictícia) e o Contato Normalmente Fechado permanece fechado (dando condição ao fluxo desta corrente)." Esta definição é resumida na tabela 4.1.

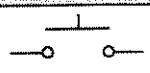
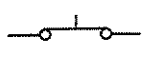
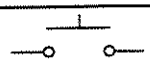
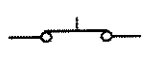
DISPOSITIVO DE ENTRADA	TABELA DE IMAGEM DAS ENTRADAS	ELEMENTO DA LÓGICA DE CONTROLE	ATUAÇÃO DO CONTATO LÓGICO	FLUXO DE CORRENTE LÓGICA
	0		NÃO	NÃO
	1		SIM	SIM
	0		NÃO	SIM
	1		SIM	NÃO

Tabela 4.1 – Relação Dispositivo de Entrada x Elemento da Lógica de Controle

Conforme visto, independente das características do dispositivo conectado ao Módulo de Entrada (Contato Normalmente Aberto–NA ou Normalmente Fechado–NF) a Lógica de Controle pode ser implementada com Contatos NA e/ou NF referenciados ao mesmo Ponto de Entrada. Por exemplo, a Lógica de Controle implementada na figura 4.12 determina que se X0 = 1 (entrada atuada) será acionada a Saída Y0. Caso contrário, se X0 = 0 (entrada não atuada) será

acionada a Saída Y1. Embora a cada Ponto de Entrada, no caso X0, possa ser conectado apenas um tipo de contato do dispositivo de entrada (NA ou NF), a Lógica de Controle pode ser implementada de tal forma que realize operações distintas, conforme a atuação ou não do dispositivo de entrada, no caso acionando Y0 ou Y1.

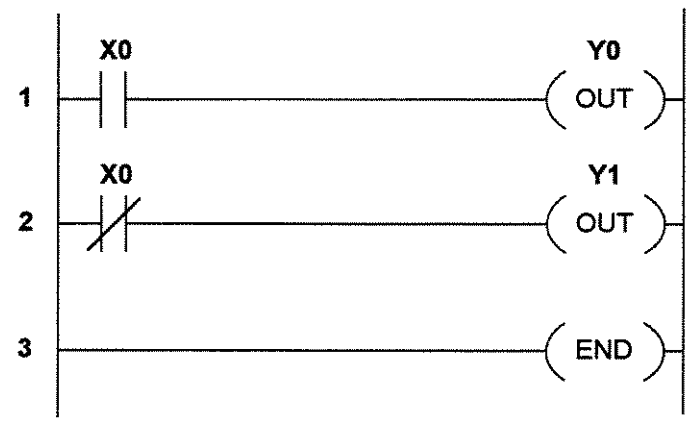


Figura 4.12 – Utilização de Contatos NA e NF Referenciados ao Mesmo Ponto de Entrada

Cada Ponto de Entrada tem apenas um único endereço a ele relacionado (X0, X1, ...), porém pode ser utilizado tantas vezes quantas forem necessárias para a implementação da Lógica de Controle, ora como Contato NA, ora como Contato NF, tendo como único limite a quantidade de memória disponível ao armazenamento do Programa de Aplicação.

4.2 Tipos De Dados

Além dos Pontos de Entrada e Saída Discretas há outros Elementos (ou Tipos de Dados) utilizados na implementação da Lógica de Controle. Embora cada PLC utilize nomenclatura, representação gráfica (Linguagem Ladder) e forma de endereçamento próprias, a equivalência entre os Tipos de Dados disponíveis em CPUs distintas proporciona rápida adaptação ao usuário.

Os Tipos de Dados, apresentados a seguir, referem-se às CPUs *PLC Direct by Koyo*. Porém, os mesmos estão presentes, com as possíveis diferenças citadas acima, na maioria dos PLCs disponíveis no mercado.

4.2.1 Entradas Discretas – Tipo de Dado: X

As Entradas Discretas são identificadas por **X** (Dado Tipo X), e cada Ponto é endereçado em base octal (X0, X1, X2,..., X7, X10, X11,..., X77, X100, X101,...). Normalmente, estão associadas às Instruções Booleanas de Entrada (Contatos NA ou NF).

4.2.2 Saídas Discretas – Tipo de Dado: Y

As Saídas Discretas são identificadas por **Y** (Dado Tipo Y), e cada Ponto é endereçado em base octal (Y0, Y1, Y2,..., Y7, Y10,...). Embora estejam, normalmente, associadas às Instruções Boolenas de Saída (Bobinas de diversas funções), podem ser utilizadas também em Instruções Boolenas de Entrada (Contatos NA ou NF), conforme a necessidade.

Na Lógica de Controle implementada na figura 4.13, pode-se observar a utilização do Tido de Dado Y (Y0) associado a uma Instrução de Entrada (Contato NF). Neste caso, a Saída Y0 é acionada a partir da condição das Entradas X0 e X1 (**X0=1** e **X1=0**). Caso esta condição não seja satisfeita, a Saída Y0 não é acionada (mantendo-se desligada), ocasionando o acionamento da Saída Y1 (**Y0=0**).

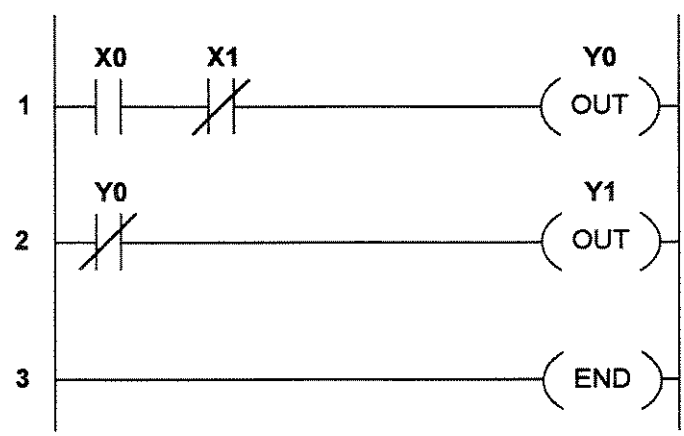


Figura 4.13 – Utilização de Entradas e Saídas Discretas

4.2.3 Relés de Controle – Tipo de Dado: C

Tratam-se de *bits* internos à CPU utilizados como Relés de Controle (ou Auxiliares), não tendo conexão a dispositivos externos de entrada ou saída. São identificados por **C** (Dado Tipo C), e cada Relé de Controle é endereçado em base octal (C0, C1, C2,..., C7, C10,...). São

associados às Instruções Booleanas de Entrada ou de Saída. A Lógica de Controle implementada na figura 4.14 exemplifica a utilização dos Relés de Controle.

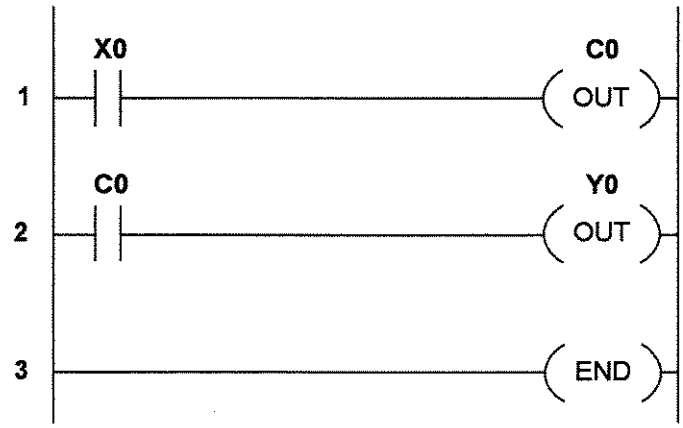


Figura 4.14 – Utilização de Relé de Controle

4.2.4 Temporizadores (*Timers*) e *Bits de Status* – Tipo de Dado: T

Normalmente, são utilizados para temporização de condições e/ou eventos controlados pelo Programa de Aplicação. Cada Temporizador é identificado por **T** (Dado Tipo T), e endereçado em base octal (T0, T1, T2,...,T7, T10,...). Há um *Bit de Status* (*Bit* de Condição) relacionado a cada Temporizador (com mesmo endereço), o qual é ativado quando o Valor Atual do Temporizador for igual ou superior ao Valor de *Preset* (Valor Pré-Configurado). A razão do incremento de tempo depende do Tipo de Temporizador utilizado.

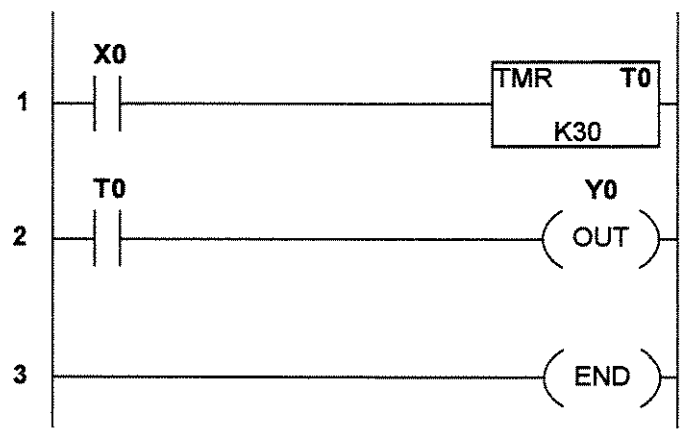


Figura 4.15 – Utilização de Temporizador e *Bit de Status*

Na Lógica de Controle implementada na figura 4.15, ao ser atuada a Entrada X0 é iniciada a temporização de T0, o qual tem Valor de *Preset* fixo em 3,0 segundos (o Temporizador apresentado possui incremento de tempo de 0,1s, portanto K30 equivale a 30 x 0,1s). Ao ser atingido o Valor de *Preset*, é ativado o *Bit de Status* de T0, acionando a Saída Y0. O *Bit de Status* de T0 permanece ativado até que o Temporizador seja desativado (**X0 = 0**, Valor Atual de T0 = 0). Este *bit* pode ser associado a um Contato NA ou NF, conforme a necessidade.

4.2.5 Valor Atual dos Temporizadores – Tipo de Dado: V (TA)

Além de monitorar se o Temporizador atingiu o Valor de *Preset*, através do *Bit de Status*, é possível ter acesso ao Valor Atual de cada Temporizador durante a execução do Programa de Aplicação, através de um endereço específico da Tabela de Dados. Por exemplo, o endereço 0 (ou V0) contém o Valor Atual de T0, o endereço 1 (ou V1) contém o Valor Atual de T1, e assim sucessivamente. Estes endereços podem ser utilizados na implementação da Lógica de Controle, proporcionando flexibilidade no desenvolvimento do Programa de Aplicação.

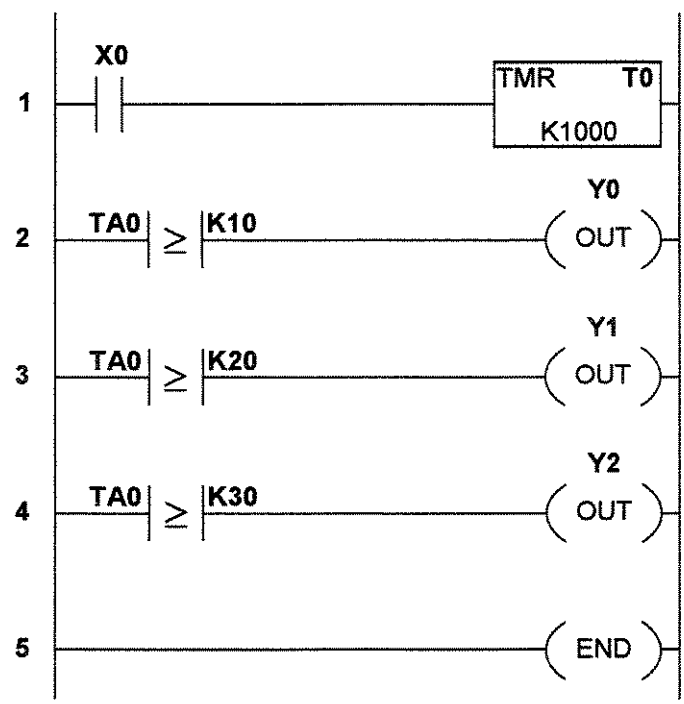


Figura 4.16 – Utilização do Valor Atual do Temporizador

O Software de Programação da *PLC Direct by Koyo* (*Direct*SOFT) permite que em vez do endereço (V0, V1, V2,...) seja utilizada uma representação mais apropriada (TA0 para T0, TA1 para T1, TA2 para T2, ...) como indicação do Valor Atual de cada Temporizador.

Na Lógica de Controle implementada na figura 4.16, ao ser atuada a Entrada X0 é iniciada a temporização de T0. Ao atingir 1s, a Saída Y0 é acionada (**Valor Atual de T0 ≥ K10**); ao atingir 2s, a Saída Y1 é acionada (**Valor Atual de T0 ≥ K20**); e ao atingir 3s, a Saída Y2 é acionada (**Valor Atual de T0 ≥ K30**).

4.2.6 Contadores (*Counters*) e *Bits de Status* – Tipo de Dado: CT

Normalmente, são utilizados para contagem de condições e/ou eventos controlados pelo Programa de Aplicação. Cada Contador é identificado por **CT** (Dado Tipo CT), e endereçado em base octal (CT0, CT1, CT2,...,CT7, CT10,...). Há um *Bit de Status* (*Bit* de Condição) relacionado a cada Contador (com mesmo endereço), o qual é ativado quando o Valor Atual do Contador for igual ou superior ao Valor de *Preset* (Valor Pré-Configurado).

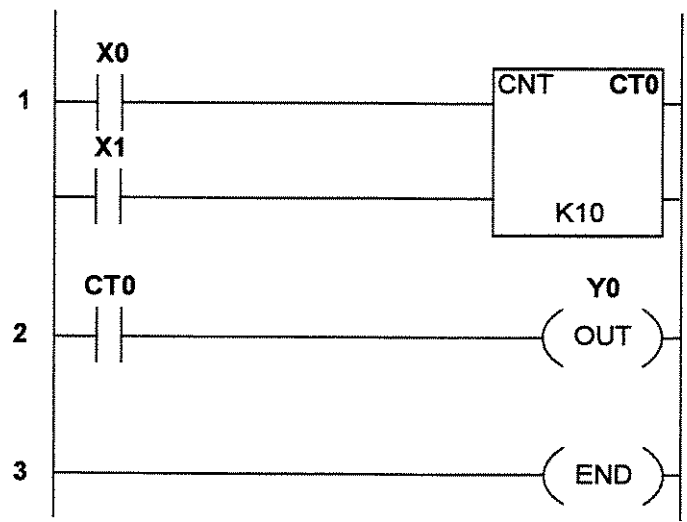


Figura 4.17 – Utilização de Contador e *Bit de Status*

Na Lógica de Controle implementada na figura 4.17, a cada transição de 0 para 1 (*off* → *on*) da Entrada X0, o Valor Atual de CT0 é incrementado em uma unidade, o qual tem Valor de *Preset* fixo em 10. Ao ser atingido o Valor de *Preset*, é ativado o *Bit de Status* de CT0, acionando

a Saída Y0. O *Bit de Status* de CT0 permanece ativado até que o Contador seja reinicializado (*resetado*) através da atuação da Entrada X1 (Valor Atual de CT0 = 0). Este *bit* pode ser associado a um Contato NA ou NF, conforme a necessidade.

4.2.7 Valor Atual dos Contadores – Tipo de Dado: V (CTA)

Além de monitorar se o Contador atingiu o Valor de *Preset*, através do *Bit de Status*, é possível ter acesso ao Valor Atual de cada Contador durante a execução do Programa de Aplicação, através de um endereço específico da Tabela de Dados. Por exemplo, o endereço 1000 (ou V1000) contém o Valor Atual de CT0, o endereço 1001 (ou V1001) contém o Valor Atual de CT1, e assim sucessivamente. Estes endereços podem ser utilizados na implementação da Lógica de Controle, proporcionando flexibilidade no desenvolvimento do Programa de Aplicação.

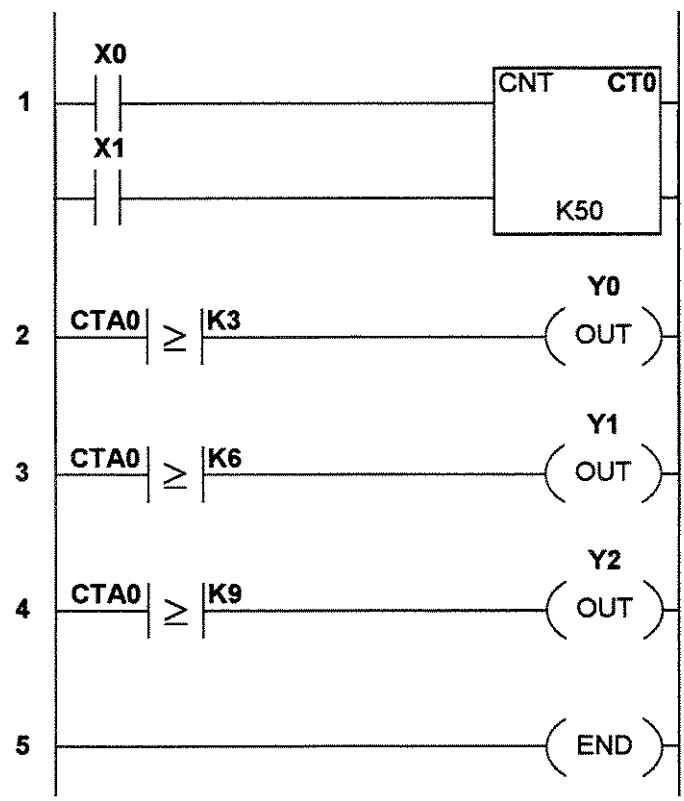


Figura 4.18 – Utilização do Valor Atual do Contador

O Software de Programação da *PLC Direct by Koyo* (*Direct*SOFT) permite que em vez do endereço (V1000, V1001, V1002,...) seja utilizada uma representação mais apropriada (CTA0

para CT0, CTA1 para CT1, CTA2 para CT2,...) como indicação do Valor Atual de cada Contador.

Na Lógica de Controle implementada na figura 4.18, a cada transição de 0 para 1 da Entrada X0, o Valor Atual de CT0 é incrementado em uma unidade. Ao atingir o valor de contagem igual a 3, a Saída Y0 é acionada (**Valor Atual de CT0 ≥ K3**); ao atingir o valor de contagem igual a 6, a Saída Y1 é acionada (**Valor Atual de CT0 ≥ K6**); e ao atingir o valor de contagem igual a 9, a Saída Y2 é acionada (**Valor Atual de CT0 ≥ K9**).

4.2.8 Variáveis (*Words*) – Tipo de Dado: V

São posições de memória de 16 *bits* geralmente utilizadas para armazenamento ou manipulação de dados e valores. Cada *Word* é identificada por **V** (Dado Tipo V), e endereçada em base octal (V2000, V2001,..., V2007, V2010,...). Alguns endereços tem funções pré-definidas, como por exemplo V0 (armazena o Valor Atual de T0) ou V1000 (armazena o Valor Atual de CT0), sendo que a área livre disponível para o usuário é determinada pela CPU utilizada.

Todas as CPUs *PLC Direct by Koyo* possuem acumulador de 32 *bits*. Assim, quando necessário, pode-se utilizar Instruções Duplas (que manipulam 32 *bits*) para operar com duas *Words* consecutivas de memória (V2000 e V2001, por exemplo).

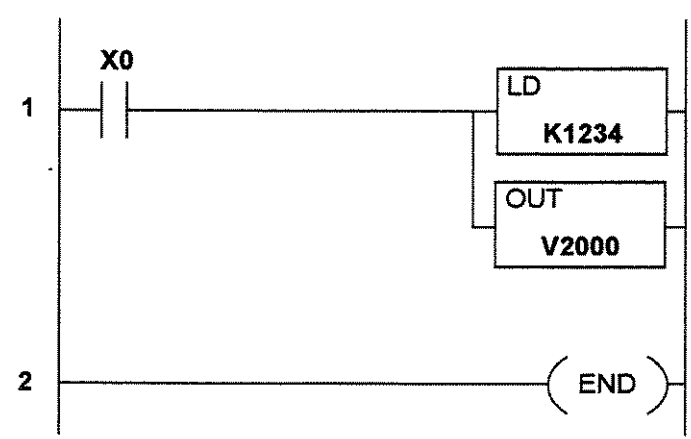


Figura 4.19 – Utilização de Variável (*Word*)

Na Lógica de Controle implementada na figura 4.19, ao ser atuada a Entrada X0, o valor 1234 (em formato BCD/HEX) é carregado no acumulador (LD / *Load* – Carregar) e então enviado para a *Word* V2000 (OUT).

4.2.9 Estágios – Tipo de Dado: S

Os Estágios são usados no desenvolvimento de Programas de Aplicação estruturados, algo semelhante ao *SFC – Sequential Function Chart* ou ao *Grafcet*, através de Instruções RLL^{PLUS} (*Relay Ladder Logic Plus*) disponível na maioria das CPUs *PLC Direct by Koyo*.

São identificados por **S** (Dado Tipo S), e endereçados em base octal (S0, S1,..., S7, S10,...). Cada Estágio representa um segmento do Programa de Aplicação, e contém a Lógica de Controle referente a este, que será executada apenas se o Estágio estiver ativo. Caso contrário, a CPU 'salta' para o próximo Estágio que atenda a esta condição. Há também um *Bit de Status* (*Bit* de Condição) relativo a cada Estágio (com mesmo endereço), que pode ser associado às Instruções Booleanas de Entrada (para indicar se o Estágio está ativo ou não – Contatos NA ou NF) ou de Saída (para determinar o acionamento ou não do Estágio – Instruções de *Jump*, *Set* ou *Reset*).

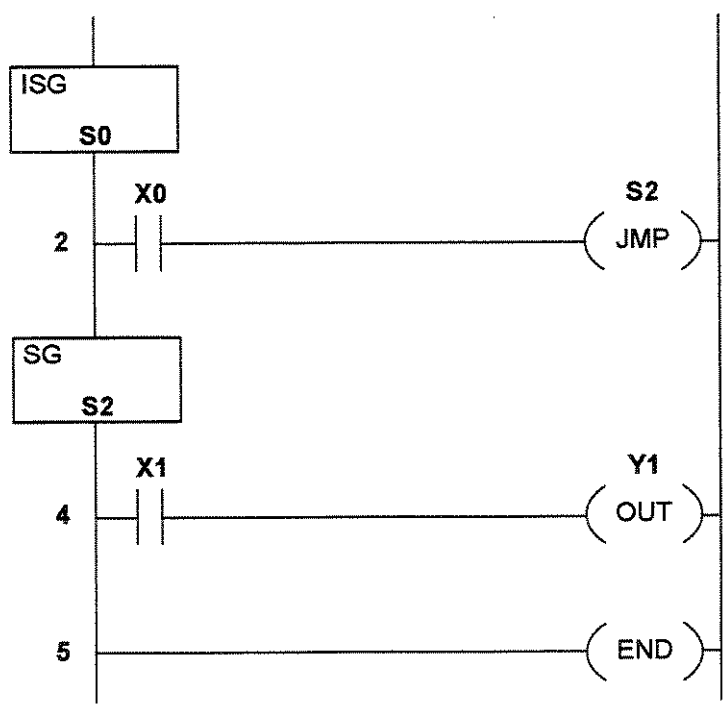


Figura 4.20 – Utilização de Estágios

A Lógica de Controle implementada na figura 4.20 determina que, ao iniciar a execução do Programa de Aplicação, mesmo que a Entrada X1 seja atuada (**X1 = 1**), a Saída Y1 não será acionada, pois o Estágio que contém a Lógica de Controle (S2) não estará ativo. Neste momento, apenas o Estágio Inicial (S0) estará e permanecerá ativo, até que a Entrada X0 seja atuada (**X0 = 1**), dando condição de *Jump* para o Estágio S2, o que ocasiona a desativação de S0 e a ativação de S2 (**S0 = 0** e **S2 = 1**), permitindo que a Entrada X1 acione a Saída Y1.

4.2.10 Relés Especiais (*Special Relays*) – Tipo de Dado: SP

Tratam-se de *bits* internos à CPU com funções pré-definidas pelo fabricante, como Indicação de *Status* do Acumulador da CPU, Monitoramento do Sistema, Indicação de Erros e Base de Tempo-Real, por exemplo.

São identificados por **SP** (Dado Tipo SP), endereçados em base octal e só podem ser associados às Instruções Booleanas de Entrada (Contatos NA ou NF). A tabela 4.2 apresenta alguns exemplos de Relés Especias.

RELÉ	FUNÇÃO	DESCRIÇÃO
SP0	Primeiro Scan	Ativado apenas no primeiro *scan* da CPU; desativado nos demais *scans*
SP1	Sempre ON	Ativado em todos os *scans* da CPU
SP2	Sempre OFF	Desativado em todos os *scans*
SP3	Clock 1 Minuto	30 segundos *on* e 30 segundos *off*
SP4	Clock 1 Segundo	0,5 segundo *on* e 0,5 segundo *off*
SP5	Clock 100 ms	50 ms *on* e 50 ms *off*
SP6	Clock 50 ms	25 ms *on* e 25ms *off*
SP7	Scan Alternados	Ativo em um *scan* e desativado no *scan* seguinte, alternadamente

Tabela 4.2 – Relés Especiais (Funções de *Start-Up* e Base de Tempo-Real)

Na Lógica de Controle implementada na figura 4.21, ao ser atuada a Entrada X0, a Saída Y0 permanecerá acionada por 0,5s e desligada por 0,5s (tempo determinado por SP4).

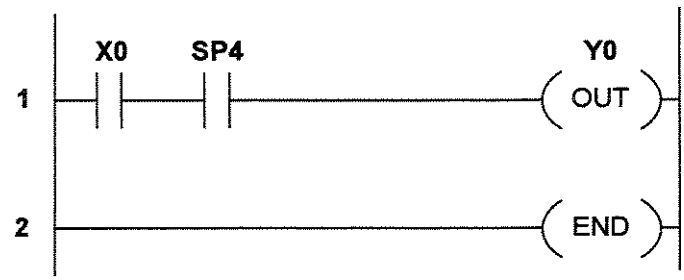


Figura 4.21 – Utilização de Relés Especiais

4.2.11 Mapeamento de Memória

Todos os Tipos de Dados apresentados têm uma área de memória (*Word* ou *V-Memory*) reservada para este fim. A quantidade de cada um dos Tipos de Dados disponíveis depende da CPU utilizada, porém o endereço inicial destas áreas (Mapeamento) é sempre o mesmo, conforme apresentado nas tabelas 4.3, 4.4, 4.5, 4.6, 4.7 e 4.8.

MSB					Relação Word.bit x Dado Tipo X											LSB	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
X17	X16	X15	X14	X13	X12	X11	X10	X7	X6	X5	X4	X3	X2	X1	X0	V40400
X37	X36	X35	X34	X33	X32	X31	X30	X27	X26	X25	X24	X23	X22	X21	X20	V40401
X57	X56	X55	X54	X53	X52	X51	X50	X47	X46	X45	X44	X43	X42	X41	X40	V40402
X77	X76	X75	X74	X73	X72	X71	X70	X67	X66	X65	X64	X63	X62	X61	X60	V40403

Tabela 4.3 – Endereçamento das Entradas Discretas – Dado Tipo X (Endereços Iniciais)

MSB					Relação Word.bit x Dado Tipo Y											LSB	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	V40500
Y37	Y36	Y35	Y34	Y33	Y32	Y31	Y30	Y27	Y26	Y25	Y24	Y23	Y22	Y21	Y20	V40501
Y57	Y56	Y55	Y54	Y53	Y52	Y51	Y50	Y47	Y46	Y45	Y44	Y43	Y42	Y41	Y40	V40502
Y77	Y76	Y75	Y74	Y73	Y72	Y71	Y70	Y67	Y66	Y65	Y64	Y63	Y62	Y61	Y60	V40503

Tabela 4.4 – Endereçamento das Saídas Discretas – Dado Tipo Y (Endereços Iniciais)

MSB					Relação Word.bit x Dado Tipo C											LSB	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
C17	C16	C15	C14	C13	C12	C11	C10	C7	C6	C5	C4	C3	C2	C1	C0	V40600
C37	C36	C35	C34	C33	C32	C31	C30	C27	C26	C25	C24	C23	C22	C21	C20	V40601
C57	C56	C55	C54	C53	C52	C51	C50	C47	C46	C45	C44	C43	C42	C41	C40	V40602
C77	C76	C75	C74	C73	C72	C71	C70	C67	C66	C65	C64	C63	C62	C61	C60	V40603

Tabela 4.5 – Endereçamento dos Relés de Controle – Dado Tipo C (Endereços Iniciais)

MSB					Relação Word.bit x Dado Tipo T											LSB	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
T17	T16	T15	T14	T13	T12	T11	T10	T7	T6	T5	T4	T3	T2	T1	T0	V41100
T37	T36	T35	T34	T33	T32	T31	T30	T27	T26	T25	T24	T23	T22	T21	T20	V41101
T57	T56	T55	T54	T53	T52	T51	T50	T47	T46	T45	T44	T43	T42	T41	T40	V41102
T77	T76	T75	T74	T73	T72	T71	T70	T67	T66	T65	T64	T63	T62	T61	T60	V41103

Tabela 4.6 – Endereçamento dos *Bits de Status* dos Temporizadores – Dado Tipo T (Endereços Iniciais)

MSB					Relação Word.bit x Dado Tipo CT											LSB	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
CT17	CT16	CT15	CT14	CT13	CT12	CT11	CT10	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0	V41400
CT37	CT36	CT35	CT34	CT33	CT32	CT31	CT30	CT27	CT26	CT25	CT24	CT23	CT22	CT21	CT20	V41401
CT57	CT56	CT55	CT54	CT53	CT52	CT51	CT50	CT47	CT46	CT45	CT44	CT43	CT42	CT41	CT40	V41402
CT77	CT76	CT75	CT74	CT73	CT72	CT71	CT70	CT67	CT66	CT65	CT64	CT63	CT62	CT61	CT60	V41403

Tabela 4.7 – Endereçamento dos *Bits de Status* dos Contadores – Dado Tipo CT (Endereços Iniciais)

MSB					Relação Word.bit x Dado Tipo S											LSB	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
S17	S16	S15	S14	S13	S12	S11	S10	S7	S6	S5	S4	S3	S2	S1	S0	V41000
S37	S36	S35	S34	S33	S32	S31	S30	S27	S26	S25	S24	S23	S22	S21	S20	V41001
S57	S56	S55	S54	S53	S52	S51	S50	S47	S46	S45	S44	S43	S42	S41	S40	V41002
S77	S76	S75	S74	S73	S72	S71	S70	S67	S66	S65	S64	S63	S62	S61	S60	V41003

Tabela 4.8 – Endereçamento dos *Bits de Status* dos Estágios – Dado Tipo S (Endereços Iniciais)

Os Dados Tipo TA (Valor Atual dos Temporizadores) têm endereço inicial em V0 e os Dados Tipo CTA (Valor Atual dos Contadores) têm endereço inicial em V1000, conforme apresentado anteriormente. Já os Dados Tipo V (*Word*) têm endereço inicial dependente da CPU utilizada. Porém, o endereço V2000 é referência para a maioria dos modelos.

4.3 Programação em Linguagem Ladder (RLL – *Relay Ladder Logic*)

Atualmente, os PLCs apresentam Instruções sofisticadas. Além de simples contatos e bobinas, dispõem de contatos para detecção de borda de subida/descida (*one shot* – disparo), contatos de comparação, temporizadores, contadores, blocos de processamento (operações lógicas e aritméticas, manipulação de dados e tabelas), controle total do fluxo de execução do programa (*loops For/Next*, *Goto*, *Stop*, sub-rotinas) e interrupções (por hardware e software), por exemplo.

A quantidade e diversidade das Instruções disponíveis em cada PLC estão diretamente relacionadas à CPU utilizada. Por exemplo, a Família DL205 da *PLC Direct by Koyo* oferece atualmente três modelos de CPUs – DL230, DL240 e DL250. A CPU DL230 é o modelo básico, a DL240 o intermediário e a DL250 o mais sofisticado. A CPU DL230 não possui as Instruções *For/Next* (para construção de *loops*); a CPU DL240 possui tais Instruções, porém as Instruções Aritméticas só operam com números inteiros; já a CPU DL250, além de possuir as Instruções *For/Next*, possui Instruções Aritméticas que operam com números reais (ponto flutuante). No entanto, as diferenças entre estas CPUs não se limitam apenas ao Conjunto de Instruções.

As Instruções podem ser classificadas em categorias conforme a função executada: Instruções Booleanas, Instruções de Temporização, Instruções de Contagem, Instruções de Controle do Programa, Instruções de Manipulação de Dados, Instruções Lógicas, Instruções Aritméticas e Instruções Especiais, por exemplo. As possíveis diferenças de classificação, nomenclatura e representação entre PLCs distintos são, normalmente, assimiladas com facilidade pelos usuários.

A seguir são apresentadas as Instruções mais comuns da Linguagem Ladder, referentes às CPUs *PLC Direct by Koyo* e presentes na grande maioria dos PLCs disponíveis no mercado.

4.3.1 Instruções Booleanas de Entrada

As Instruções Booleanas de Entrada geralmente resumem-se em Contato Normalmente Aberto (NA) e Contato Normalmente Fechado (NF). Porém, a localização destes na Lógica de Controle (*rung*) define Instruções distintas – conforme verificado em Linguagem de Lista de Instruções na figura 4.3.

Os seguintes Tipos de Dados são válidos, para todas as CPUs *PLC Direct by Koyo*, como operandos das Instruções apresentadas:

X – Entrada Discreta
Y – Saída Discreta
C – Relé de Controle
S – Estágio
T – *Bit de Status* do Temporizador
CT – *Bit de Status* do Contador
SP – Relé Especial

Algumas CPUs aceitam o *bit* de uma determinda *Word* como operando, ou seja, Dado Tipo **V.b**, no qual **b** indica o *bit* da *Word* **V** (V2000.0 → *bit* 0 da *Word* V2000).

Store (STR)

A Instrução *Store* inicia um novo *rung* ou uma associação paralela (*branch*) adicional de um *rung* com um Contato NA, conforme apresentado na figura 4.22.

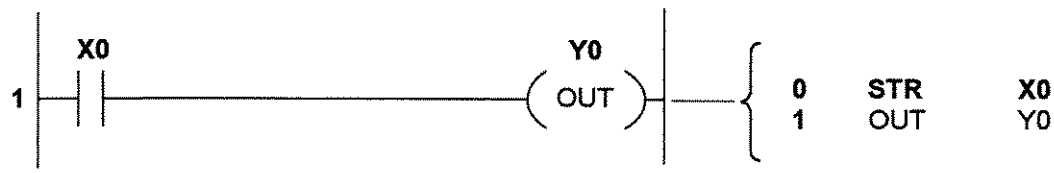


Figura 4.22 – Instrução *Store*

Store Not (STRN)

A Instrução *Store Not* inicia um novo *rung* ou uma associação paralela (*branch*) adicional de um *rung* com um Contato NF, conforme apresentado na figura 4.23.

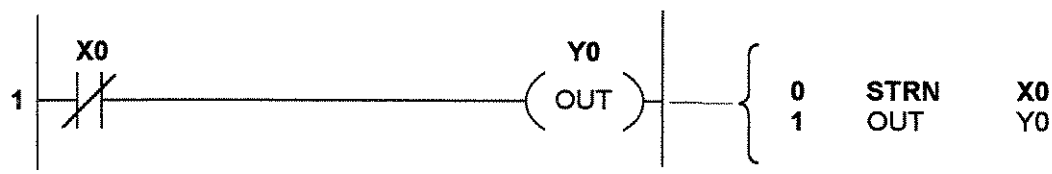


Figura 4.23 – Instrução *Store Not*

Or (OR)

A Instrução *Or* executa a Lógica OR (**A** + **B**) entre um Contato NA em paralelo a outro Contato qualquer (NA ou NF) em um *rung*, conforme apresentado na figura 4.24.

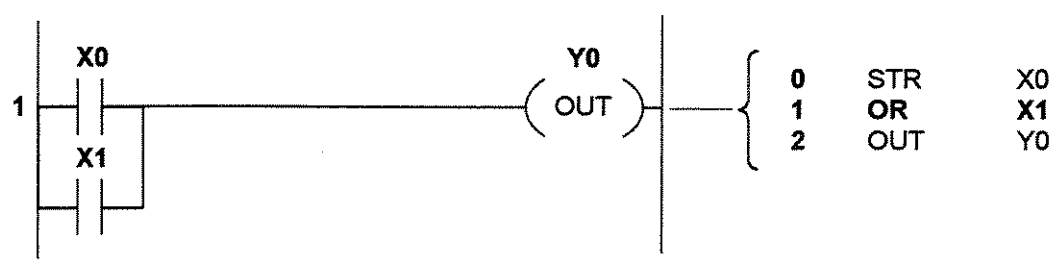


Figura 4.24 – Instrução *Or*

Or Not **(ORN)**

A Instrução *Or Not* executa a Lógica OR (**A + B**) entre um Contato NF em paralelo a outro Contato qualquer (NA ou NF) em um *rung*, conforme apresentado na figura 4.25.

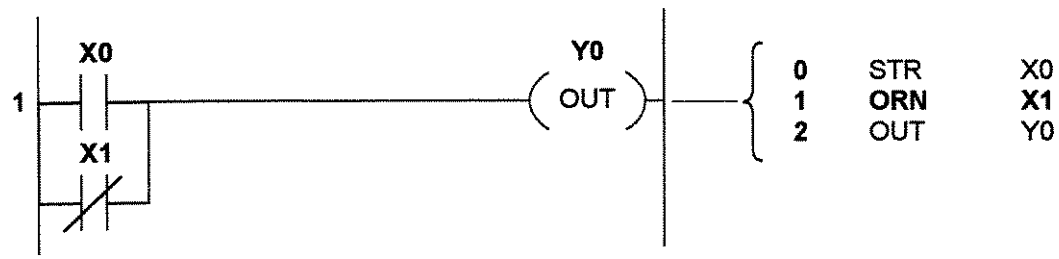


Figura 4.25 – Instrução *Or Not*

And **(AND)**

A Instrução *And* executa a Lógica AND (**A . B**) entre um Contato NA em série a outro Contato qualquer (NA ou NF) em um *rung*, conforme apresentado na figura 4.26.

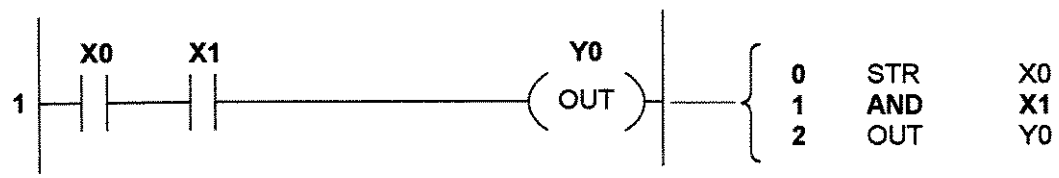


Figura 4.26 – Instrução *And*

And Not **(ANDN)**

A Instrução *And Not* executa a Lógica AND (**A . B**) entre um Contato NF em série a outro Contato qualquer (NA ou NF) em um *rung*, conforme apresentado na figura 4.27.

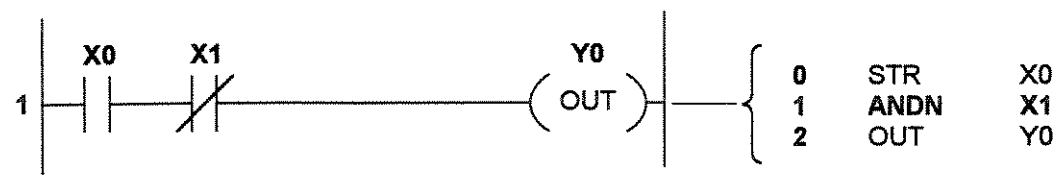


Figura 4.27 – Instrução *And Not*

Or Store (OR STR)

A Instrução *Or Store* executa a lógica OR (**A + B**) entre duas associações paralelas (*branches*) de um mesmo *rung*, em paralelo. Ambas as seções devem iniciar com a Instrução *Store* ou *Store Not*, conforme apresentado na figura 4.28.

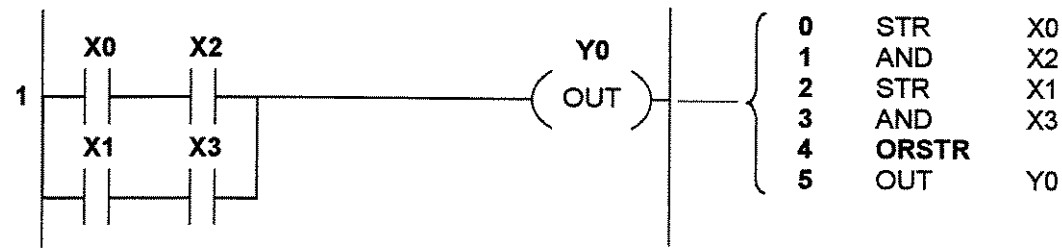


Figura 4.28 – Instrução *Or Store*

And Store (AND STR)

A Instrução *And Store* executa a lógica AND (**A . B**) entre duas associações paralelas (*branches*) de um mesmo *rung*, em série. Ambas as seções devem iniciar com a Instrução *Store* ou *Store Not*, conforme apresentado na figura 4.29.

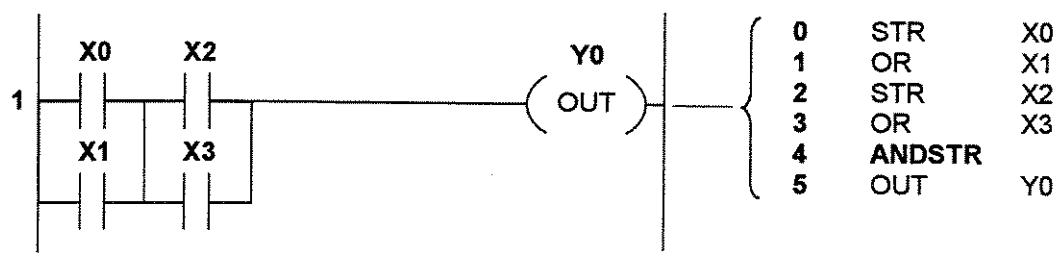


Figura 4.29 – Instrução *And Store*

4.3.2 Instruções Booleanas de Saída

As Instruções Booleanas de Saída são representadas por Bobinas, e diferenciam-se pelas funções executadas (*Out*, *Set* e *Reset*, por exemplo).

Os seguintes Tipos de Dados são válidos, para todas as CPUs *PLC Direct by Koyo*, como operandos das Instruções apresentadas:

X – Entrada Discreta **Y** – Saída Discreta **C** – Relé de Controle

Algumas CPUs aceitam o *bit* de uma determinda *Word* como operando, ou seja, Dado Tipo **V.b**, no qual **b** indica o *bit* da *Word* **V** (V2000.0 → *bit* 0 da *Word* V2000).

Embora seja possível utilizar o Dado Tipo X (Entrada Discreta) como operando de uma Instrução Booleana de Saída, é recomendado fazê-lo apenas para os endereços não utilizados pelos Módulos de Entrada – Discretos ou Analógicos (neste caso, os Dados Tipo X teriam a mesma função dos Relés de Controle – Dado Tipo C).

***Out* (OUT)**

A Instrução *Out* reflete o *status* do *rung* (*on/off*) sobre o operando controlado. No exemplo apresentado na figura 4.30, se a Entrada X0 estiver atuada (**X0 = 1**), permitindo o fluxo de Corrente Lógica Fictícia (*rung = on*), a Saída Y0 será acionada. Caso contrário (**X0 = 0** e *rung = off*), a Saída Y0 permanecerá desligada.

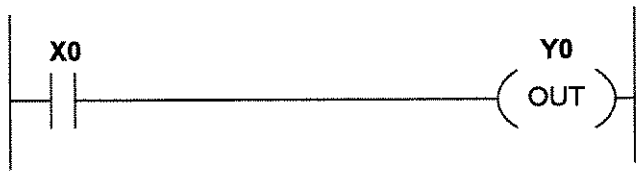


Figura 4.30 – Instrução *Out*

Devido à seqüência do Sistema de Operação da CPU (Leitura das Entradas → Execução do Programa de Aplicação – da esquerda para direita, de cima para baixo → Escrita das Saídas), não é recomendada a utilização de múltiplas Instruções *Out* referenciadas ao mesmo operando (principalmente se for uma Saída Discreta), pois apenas a última Instrução atuará sobre a Saída controlada. As Instruções anteriores atuarão apenas sobre a Tabela de Imagem das Saídas, não refletindo diretamente no Ponto de Saída controlado.

Na Lógica de Controle implementada na figura 4.31, a Saída Y0 é controlada por três condições distintas (X0, X1 e X2), associadas à Instrução *Out*. As Entradas X0 (*rung* 1) e X2 (*rung* 23) têm controle apenas momentâneo e interno (Tabela de Imagem das Saídas) sobre a Saída Y0. A Entrada X2, por ser a última condição associada à Saída Y0 (*rung* 57), tem controle real sobre o Ponto de Saída Y0, independente das demais entradas.

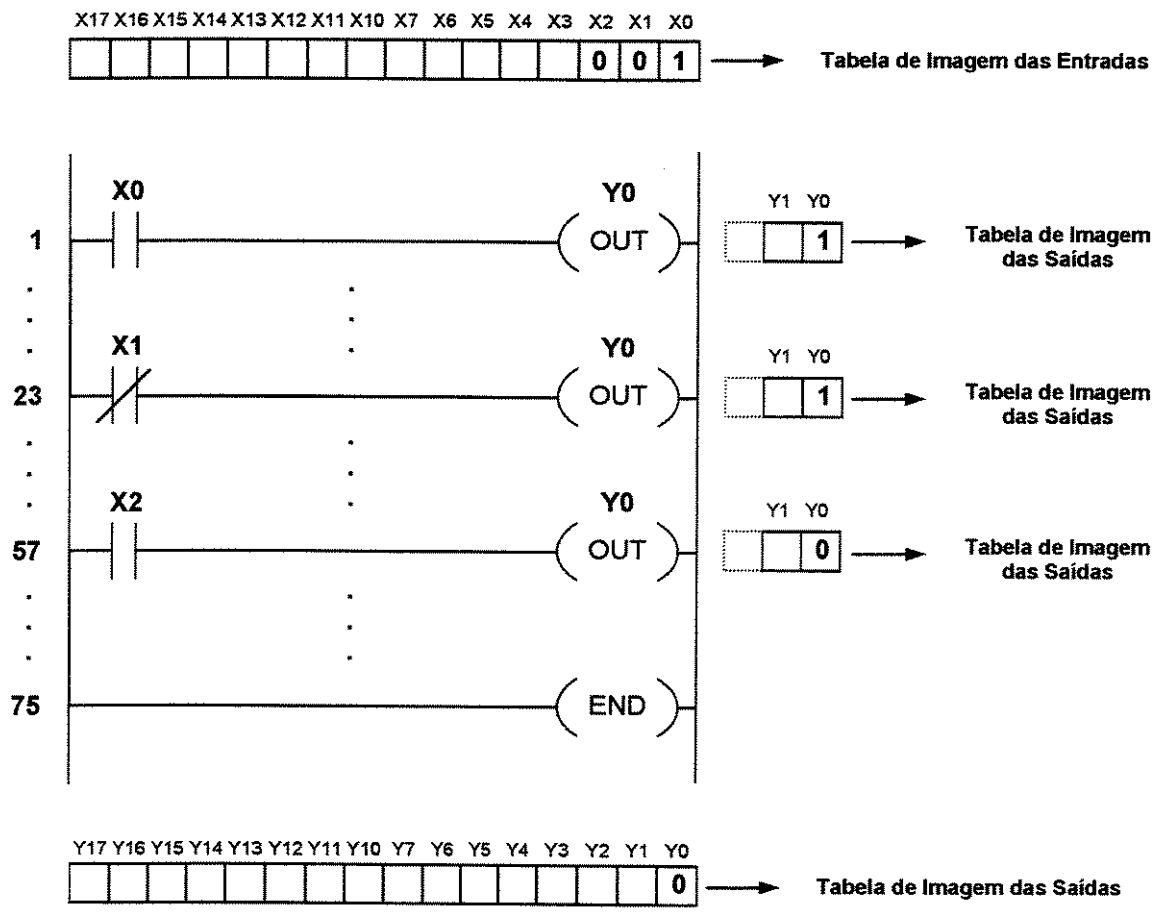


Figura 4.31 – Utilização de Múltiplas Instruções *Out*

Alguns Softwares de Programação não permitem, durante a edição do Programa de Aplicação, a utilização de múltiplas Instruções *Out* referenciadas à mesma Saída. O Software *Direct*SOFT, da *PLC Direct by Koyo*, permite tal utilização, uma vez que este recurso é muito útil na Programação por Estágios (Instruções RLL^{PLUS}). No entanto, proporciona a opção de verificação desta condição durante a edição do Programa de Aplicação.

Or Out (OROUT)

A Instrução *Or Out* tem função semelhante à Instrução *Out*, refletindo o *status* do *rung* (*on/off*) sobre o operando controlado. Porém, não tem a limitação da Instrução *Out*, podendo apresentar múltiplas referências ao mesmo operando. Se qualquer um dos *rungs* acionar a Saída controlada (*rung* = *on*), o Ponto de Saída será acionado ao final da Execução do Programa de Aplicação (Escrita das Saídas). O Ponto de Saída apenas permanecerá desligado se nenhum dos

rungs que o controlam der condição (todos os *rungs* = *off*). Ou seja, é executada a Lógica OR entre as Instruções de Saída.

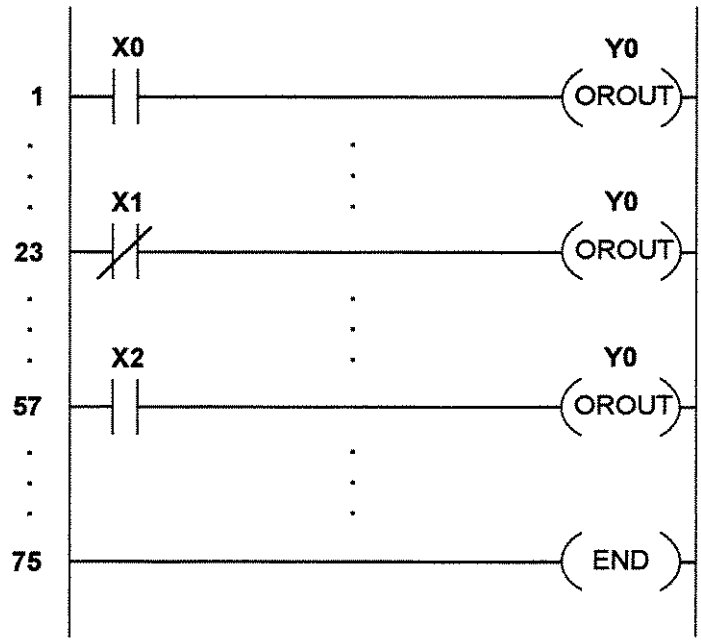


Figura 4.32 – Instrução *Or Out*

Na Lógica de Controle implementada na figura 4.32, se qualquer um dos rungs que controlam a Saída Y0 estiver atuado (**X0 = 1** e *rung* 1 = *on*; ou **X1 = 0** e *rung* 23 = *on*; ou **X2 = 1** e *rung* 57 = *on*), a Saída Y0 será acionada.

A Instrução *Or Out* está disponível em todas as CPUs *PLC Direct by Koyo*, porém não se trata de uma Instrução padrão (que esteja disponível na maioria dos PLCs).

***Positive Differential* (PD)**

A Instrução *Positive Differential*, tipicamente conhecida como *One Shot Output*, apresenta uma função especial. Quando o *rung* que controla a Instrução *Positive Differential* produzir uma transição de 0 para 1 (*off* → *on*), o operando associado à Instrução será acionado durante apenas um *scan* da CPU, permanecendo desligado até nova transição de 0 para 1 do *rung*.

Na Lógica de Controle implementada na figura 4.33, ao ser atuda a Entrada X0 (produzindo a transição de 0 para 1), o Relé de Controle C0 é acionado por um apenas *scan*, carregando valor 1234 (em formato BCD/HEX) na *Word* V2000. Independente do tempo que a Entrada X0 permanecer atuada, o valor 1234 será carregado na *Word* V2000 apenas uma vez (no primeiro *scan*), podendo ser alterado por outras condições determinadas pela Lógica de Controle, sendo recarregado a cada nova atuação da Entrada X0.

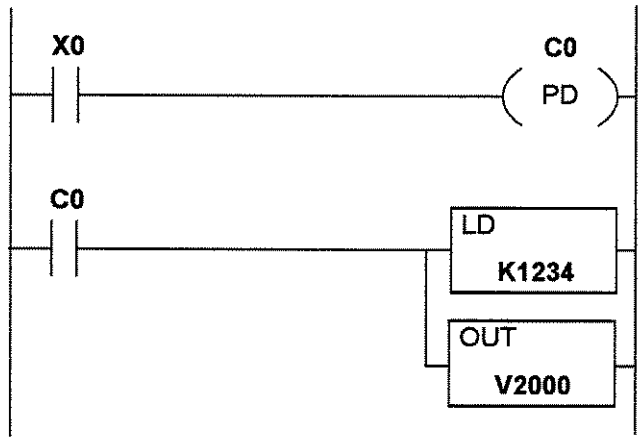


Figura 4.33 – Instrução *Positive Differential*

A Instrução *Positive Differential* está disponível em todas as CPUs *PLC Direct by Koyo*, porém não se trata de uma Instrução padrão (que esteja disponível na maioria dos PLCs). Esta Instrução é comumente utilizada quando se necessita realizar Operações Lógicas e Aritméticas, ou Manipulação de Dados, a partir da ocorrência de determinados eventos.

***Set* (SET) e *Reset* (RST)**

Ao ser executada (*rung* = 1), a Instrução *Set* aciona o operando controlado, mantendo-o nesta condição mesmo que o *rung* não permaneça acionado (*rung* = 0).

O operando acionado ('*setado*') pela Instrução *Set*, é desligado ('*resetado*') pela execução (*rung* = 1) da Instrução *Reset*, referenciada ao mesmo operando, mantendo-o nesta condição mesmo que o *rung* não permaneça acionado (*rung* = 0).

Se as Instruções *Set* e *Reset* forem executadas durante o mesmo *scan*, terá controle sobre o operando aquela que for executada por último, devido à seqüência do Sistema de Operação da CPU.

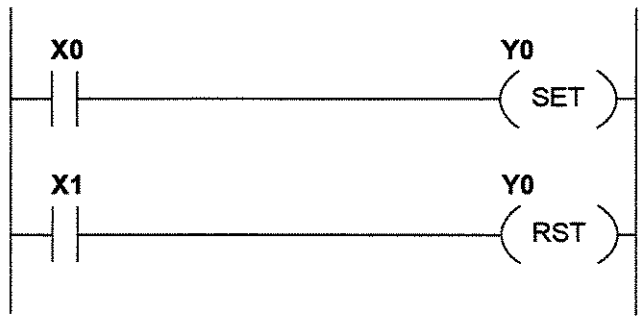


Figura 4.34 – Instruções *Set* e *Reset*

Na Lógica de Controle implementada na figura 4.34, ao ser atuada a Entrada X0 (*rung* = *on*), a Saída Y0 será '*setada*' (**Y0 = 1**), permanecendo nesta condição mesmo que a Entrada X0 não permaneça atuada (*rung* = *off*). Ao ser atuada a Entrada X1 (*rung* = *on*), a Saída Y0 será '*resetada*' (**Y0 = 0**), permanecendo nesta condição mesmo que a Entrada X1 não permaneça atuada (*rung* = *off*), até que a Instrução *Set* seja novamente executada (**X0 = 1**). Se as Entradas X0 e X1 forem atuadas simultaneamente, executando as Instruções *Set* e *Reset* no mesmo *scan*, a Saída Y0 será '*resetada*' (**Y0 = 0**), uma vez que a Instrução *Reset* será executada após a Instrução *Set*, permanecendo valor 0 na Tabela de Imagem das Saídas.

As Instruções *Set* e *Reset* podem ser referenciadas tanto a um único operando, como a uma faixa de operandos seqüenciais, conforme apresentado na figura 4.35.

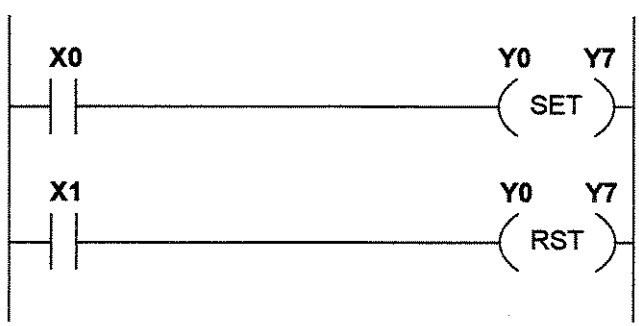


Figura 4.35 – Instruções *Set* e *Reset* Referenciadas à Faixa de Operandos

4.3.3 Instruções Booleanas de Comparação

Quando disponíveis, normalmente são quatro os tipos de Instruções Booleanas de Comparação: Igualdade, Diferença, Superioridade ou Igualdade, e Inferioridade. Por serem Instruções Boolenas de Entrada, a localização dos Contatos de Comparação na Lógica de Controle (*rung*) define Instruções distintas, porém com mesma função.

Os seguintes Operandos são válidos, para todas as CPUs *PLC Direct by Koyo*, para as Instruções apresentadas:

OPERANDO A	OPERANDO B
V – *Word*	**V** – *Word*
TA – Valor Atual do Temporizador	**TA** – Valor Atual do Temporizador
CTA – Valor Atual do Contador	**CTA** – Valor Atual do Contador
	K – Valor Constante
	P – Ponteiro de Dado

A localização, no Contato de Comparação, do Operando A é à esquerda da Função de Comparação, e do Operando B à direita, conforme apresentado na figura 4.36.

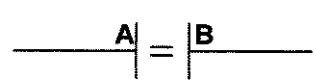


Figura 4.36 – Localização dos Operandos em Contatos de Comparação

Quando se deseja comparar o conteúdo de uma *Word* (Dado Tipo V), ou o Valor Atual de um Temporizador (Dado Tipo TA), ou o Valor Atual de um Contador (Dado Tipo CTA), com um Valor Constante, utiliza-se o Operando **Kbbbb**, onde **bbbb** é a constante em formato BCD/HEX (0-9999/0-FFFF).

Pode-se, ainda, ter como Operando B um Ponteiro de Dado (Endereçamento Indireto), que indica o endereço do Dado a ser utilizado na Comparação. Por exemplo, tendo **P3000** como Operando B, significa que o Dado (Valor) a ser utilizado na Comparação está no endereço indicado na *Word* V3000. Supondo que V3000 contenha o Dado 400_H (2000 em base octal,

portanto V2000), e que V2000 contenha o Valor 1234, o Operando A será comparado com o Valor contido na *Word* endereçada indiretamente, ou seja 1234. Isto é equivalente a ter V2000 como Operando B. Algumas CPUs *PLC Direct by Koyo* (modelos básicos de cada Família) não aceitam Ponteiros como operandos.

A seguir, são apresentados os Contatos de Comparação, independentes da localização no *rung*, sendo que os mesmos podem ser utilizados em série e/ou em paralelo a outros Elementos para a implementação da Lógica de Controle.

Comparação de Igualdade

O Contato de Comparação de Igualdade será atuado quando a condição de comparação for satisfeita (**A = B**). No *rung* apresentado na figura 4.37, a Saída Y0 será acionada apenas enquanto o Valor Atual de CT0 (CTA0) for igual a 10 (K10).

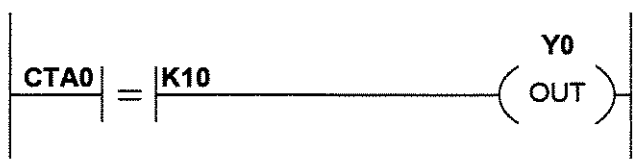


Figura 4.37 – Comparação de Igualdade

Comparação de Diferença

O Contato de Comparação de Diferença será atuado quando a condição de comparação for satisfeita (**A ≠ B**). No *rung* apresentado na figura 4.38, a Saída Y0 será acionada enquanto o Valor Atual de CT0 (CTA0) for diferente de 10 (K10).

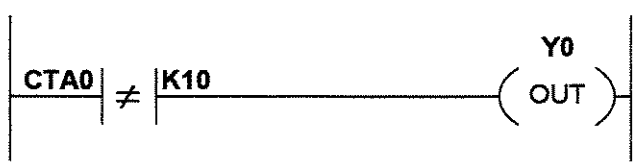


Figura 4.38 – Comparação de Diferença

Comparação de Superioridade ou Igualdade

O Contato de Comparação de Superioridade ou Igualdade será atuado quando a condição de comparação for satisfeita (**A ≥ B**). No *rung* apresentado na figura 4.39, a Saída Y0 será acionada enquanto o Valor Atual de T0 (TA0) for maior ou igual a 10 (K10).

Figura 4.39 – Comparação de Superioridade ou Igualdade

Comparação de Inferioridade

O Contato de Comparação de Inferioridade será atuado quando a condição de comparação for satisfeita (**A < B**). No *rung* apresentado na figura 4.40, a Saída Y0 será acionada enquanto o Valor Atual de T0 (TA0) for menor que 10 (K10).

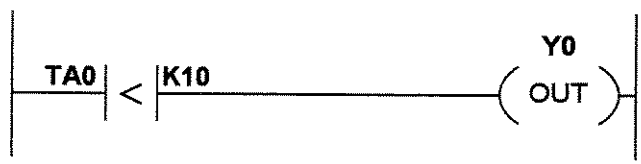


Figura 4.40 – Comparação de Inferioridade

4.3.4 Instruções Booleanas Imediatas

As Instruções Booleanas de Entrada e Saída apresentadas anteriormente, quando referenciadas às Entradas Discretas (Dados Tipo X) ou às Saídas Discretas (Dados Tipo Y) operam relacionadas à Tabela de Imagem das Entradas e à Tabela de Imagem das Saídas.

Isto significa que ao executar uma Instrução Booleana de Entrada (Contato NA ou NF) o *status* do operando (X0, por exemplo) é proveniente da Tabela de Imagem das Entradas - atualizada antes do início da Execução do Programa de Aplicação, e não do Ponto de Entrada (Módulo de I/O) naquele instante. O mesmo acontece ao executar uma Instrução Booleana de Saída, em que o *status* do operando (Y0, por exemplo) é enviado à Tabela de Imagem das Saídas,

e só ao final da Execução do Programa de Aplicação será enviado ao Ponto de Saída (Módulo de I/O).

Porém, há aplicações em que se necessita de acesso direto aos Pontos de I/O, ou seja, ao executar as Instruções Booleanas de Entrada, que o *status* do operando seja proveniente do Ponto de Entrada (Módulo de I/O) naquele instante, e ao executar Instruções Booleanas de Saída, que o *status* do operando seja enviado diretamente ao Ponto de Saída (Módulo de I/O).

Nestes casos, utilizam-se Instruções Booleanas Imediatas, que acessam diretamente os Pontos de I/O, proporcionando menor tempo de resposta. No entanto, aumentam o *Scan Time* da CPU.

As Instruções Booleanas Imediatas de Entrada resumem-se em Contatos NA e NF, sendo que a localização destes na Lógica de Controle (*rung*) define Instruções distintas, tais quais as Instruções Booleanas apresentadas anteriormente. Ao acessar diretamente o Ponto de I/O (Módulo de I/O), o *status* deste é utilizado apenas para a Instrução em questão, não realizando atualização da Tabela de Imagem das Entradas. Cada Instrução Booleana Imediata de Entrada realiza novo acesso ao Módulo de I/O, e utiliza o *status* do Ponto de I/O naquele instante para execução da Lógica de Controle. As demais Instruções Booleanas de Entrada (não-imediatas), continuam utilizando o *status* do operando proveniente da Tabela de Imagem das Entradas.

As Instruções Booleanas Imediatas de Saída são representadas por Bobinas, e diferenciam-se pelas funções executadas (*Out Immediate*, por exemplo). Ao atualizar o Ponto de I/O (Módulo de I/O) diretamente, o *status* deste é atualizado simultaneamente na Tabela de Imagem das Saídas.

A seguir, são apresentados os Contatos NA e NF Imediatos (Instruções Boolenas Imediatas de Entrada), independentes da localização no *rung*, sendo que os mesmos podem ser utilizados em série e/ou em paralelo a outros Elementos para a implementação da Lógica de Controle. O único Operando válido, em todas as CPUs *PLC Direct by Koyo*, para estas Instruções é o Dado Tipo **X** – Entrada Discreta. E também, as principais Instruções Booleanas Imediatas de Saída,

sendo que o único Operando válido, em todas as CPUs *PLC Direct by Koyo*, para estas Instruções é o Dado Tipo **Y** – Saída Discreta.

Contato NA Imediato

O *status* do Contato NA Imediato será o mesmo do Ponto de Entrada referenciado (se **X0=0** ⇒ contato aberto, se **X0=1** ⇒ contato fechado), no instante da execução da Instrução, não havendo atualização da Tabela de Imagem das Entradas. Na figura 4.41 é apresentado um Contato NA Imediato.

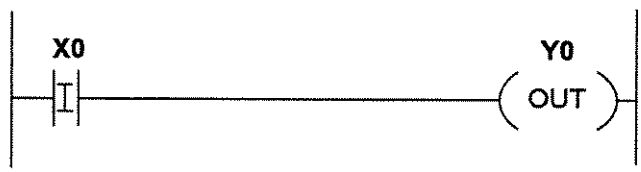


Figura 4.41 – Contato NA Imediato

Contato NF Imediato

O *status* do Contato NF Imediato será o oposto do Ponto de Entrada referenciado (se **X0=0** ⇒ contato fechado, se **X0=1** ⇒ contato aberto), no instante da execução da Instrução, não havendo atualização da Tabela de Imagem das Entradas. Na figura 4.42 é apresentado um Contato NF Imediato.

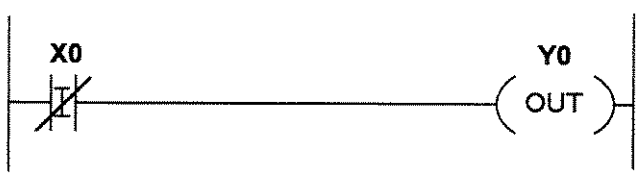


Figura 4.42 – Contato NF Imediato

***Out Immediate* (OUTI)**

A Instrução *Out Immediate* reflete o *status* do *rung* (*on/off*) sobre o Ponto de Saída controlado, no instante da execução da Instrução, atualizando também a Tabela de Imagem das Saídas. A figura 4.43 apresenta a Instrução *Out Immediate*.

Figura 4.43 – Instrução *Outi*

Se forem utilizadas múltiplas Instruções *Out Immediate* referenciadas ao mesmo Ponto de Saída, podem ocorrer alterações no *status* deste (**Y0=0** → **Y0=1** → **Y0=0** →...) durante um único *scan* da CPU, devido às Lógicas de Controle acionarem diretamente o Ponto de Saída.

A Instrução *Out Immediate* não é uma Instrução padrã (que esteja disponível na maioria dos PLCs). Algumas CPUs *PLC Direct by Koyo* não apresentam esta Instrução.

***Or Out Immediate* (OROUTI)**

A Instrução *Or Out Immediate* tem função semelhante à Instrução *Out Immediate*, refletindo o *status* do *rung* (*on/off*) sobre o Ponto de Saída controlado, no instante da execução da Instrução, atualizando também a Tabela de Imagem das Saídas.

Porém, se forem utilizadas múltiplas Instruções *Or Out Immediate* referenciadas ao mesmo Ponto de Saída, não ocorrerão as possíveis alterações de *status* durante um único *scan* da CPU, uma vez que será executada a Lógica OR entre as Instruções Imediatas de Saída. A figura 4.44 apresenta a utilização de Instruções *Or Out Immediate*.

A Instrução *Or Out Immediate* está disponível em todas as CPUs *PLC Direct by Koyo*, porém não se trata de uma Instrução padrão (que esteja disponível na maioria dos PLCs).

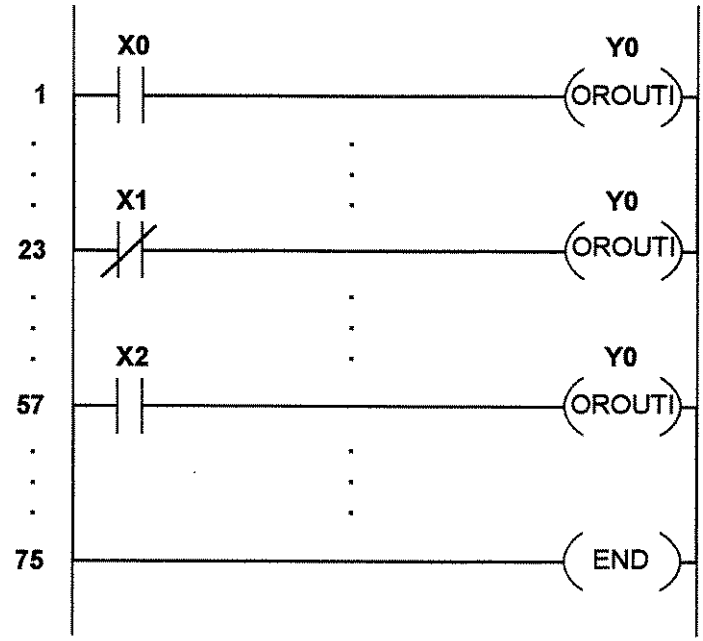


Figura 4.44 – Instrução *Or Out Immediate*

***Set Immediate* (SETI) e *Reset Immediate* (RSTI)**

No instante em que é executada (*rung* = 1), a Instrução *Set Immediate* aciona o Ponto de Saída controlado, e atualiza a Tabela de Imagem das Saídas. Uma vez executada, o Ponto de Saída permanecerá acionado mesmo que o *rung* não o permaneça (*rung* = 0).

O Ponto de Saída acionado (*'setado'*) pela Instrução *Set Immediate*, pode ser desligado (*'resetado'*) imediatamente pela execução (*rung* = 1) da Instrução *Reset Immediate*, referenciada ao mesmo Ponto de Saída. Neste instante, é atualizada a Tabela de Imagem das Saídas. Uma vez executada, o Ponto de Saída permanecerá desligado mesmo que o *rung* não o permaneça (*rung* = 0).

Não é obrigatório que um Ponto de Saída *'setado'* por uma Instrução Imediata (*Set Immediate*) seja *'resetado'* também por uma Instrução Imediata (*Reset Immediate*). O mesmo pode ser realizado por uma Instrução normal (*Reset*), e vice-versa. A figura 4.45 apresenta a utilização das Instruções *Set Immediate* e *Reset Immediate*.

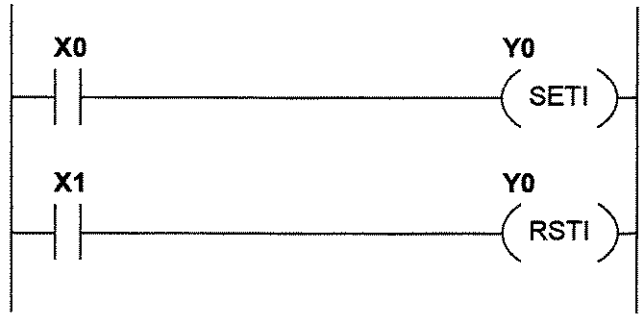


Figura 4.45 – Instruções *Set Immediate* e *Reset Immediate*

As Instruções *Set Immediate* e *Reset Immediate* podem ser referenciadas tanto a um único operando, como a uma faixa de operandos seqüenciais.

4.3.5 Temporizadores (*Timers*)

Normalmente são utilizados para temporização de condições e/ou eventos controlados pelo Programa de Aplicação. A base de tempo (razão de incremento), o tipo e a quantidade de Temporizadores disponíveis, variam conforme o fabricante e/ou a Família de PLC.

As CPUs *PLC Direct by Koyo* dispõem basicamente de dois tipos de Temporizadores: os Simples e os Acumuladores, ambos com base de tempo em décimos ou centésimos de segundo. A quantidade depende da CPU utilizada (mínimo de 64 para as CPUs mais simples). Os Temporizadores disponíveis podem ser utilizados como Simples ou Acumuladores, conforme a necessidade da Aplicação, sendo que os Acumuladores consomem dois Temporizadores (dois endereços consecutivos – T0 e T1, por exemplo) do total disponível, por operarem com oito dígitos BCD (duas *Words*).

Temporizadores Simples

- ***Timer* (TMR) / *Timer Fast* (TMRF)**

Os Temporizadores Simples possuem apenas uma Entrada de Controle – *Enable* (Habilitação), que ao ser acionada (*rung* = 1) habilita o início da temporização e ao ser desligada (*rung* = 0) reinicializa o Temporizador (Valor Atual = 0), mantendo-o nesta condição até novo acionamento da Entrada *Enable*. A figura 4.46 apresenta as especificações de um Temporizador Simples.

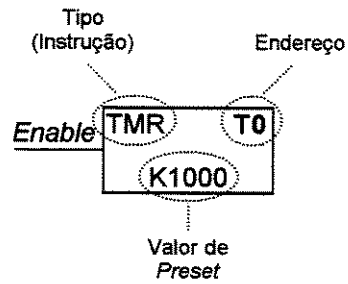


Figura 4.46 – Temporizador Simples

Especificações dos Temporizadores Simples:

- *Tipo (Instrução)*: determina a base de tempo, podendo ser **TMR** (incremento a cada décimo de segundo) ou **TMRF** (incremento a cada centésimo de segundo – ***Fast***). Ambos operam com quatro dígitos BCD (uma *Word* – 0 a 9999), sendo que o Tipo TMR temporiza de 0 a 999,9s e o Tipo TMRF de 0 a 99,99s
- *Endereço*: em base octal, possui um *Bit de Status* (*Bit* de Condição) – com mesmo endereço – relacionado, o qual é ativado quando o Valor Atual do Temporizador for igual ou superior ao Valor de *Preset* (Valor Pré-Configurado)
- *Valor de Preset*: determina o tempo que decorrerá, após a habilitação do Temporizador, para acionamento do *Bit de Status*. Os Operandos válidos como Valor de *Preset* são: **K** (Valor Constante), **V** (*Word*) e, em algumas CPUs, **P** (Ponteiro de Dados). O Valor de *Preset* deve ter formato BCD (0-9999), sendo que valores iguais podem determinar tempos diferentes, conforme o Temporizador utilizado. Por exemplo, K10 determina 1,0 segundo para Temporizadores TMR (10 x 0,1 = 1,0) e 0,1 segundo para Temporizadores TMRF (10 x 0,01 = 0,1)
- *Valor Atual*: embora não aparente na representação, o Valor Atual dos Temporizadores pode ser acessado através de *Words* específicas, sendo que V0 (ou TA0) contém o Valor Atual de T0, V1 (ou TA1) contém o Valor Atual de T1, e assim sucessivamente. Estas *Words* podem ser utilizadas na implementação da Lógica de Controle

Uma vez acionada a Entrada *Enable* (*rung* = 1), o Temporizador Simples terá seu Valor Atual incrementado conforme a base de tempo (a cada 0,1s para TMR, e a cada 0,01s para TMRF) até que a mesma seja desligada (*rung* = 0), levando o Valor Atual do Temporizador a zero. Porém, se a Entrada *Enable* permanecer habilitada até ser atingido o valor máximo de temporização (999,9s para TMR, e 99,99s para TMRF), o Valor Atual do Temporizador não mais será incrementado (não retornando a zero automaticamente – *looping*), permanecendo neste valor até que o mesmo seja reinicializado.

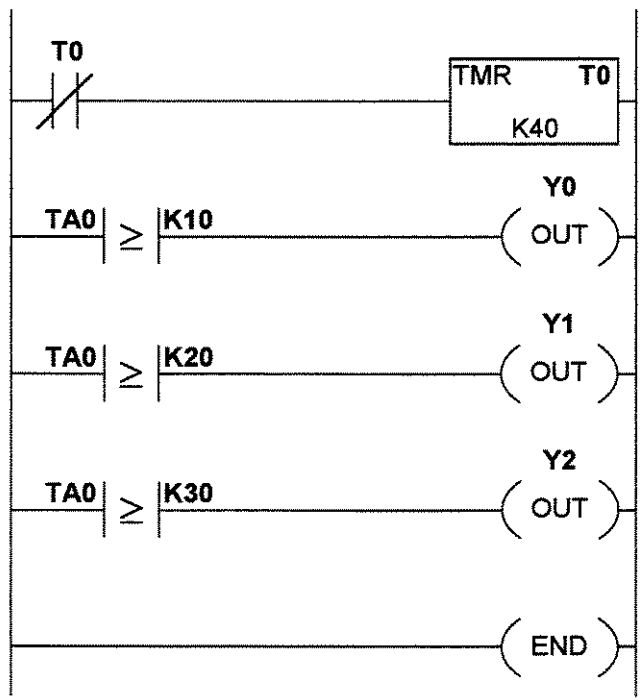


Figura 4.47 – Temporizador Simples, *Bit de Status* e Valor Atual

Na Lógica de Controle implementada na figura 4.47, tem-se um Temporizador Simples (T0) com incremento do Valor Atual (TA0) a cada 0,1s (Tipo TMR), e com Valor de *Preset* igual a 4,0s (K40). Portanto, ao iniciar a execução desta Lógica de Controle, o *Bit de Status* de T0 não está ativado (**TA0 = 0** e menor que o Valor de *Preset*), habilitando o início da temporização. Ao ser atingido o tempo de 1,0s, a Saída Y0 será acionada (**TA0 ≥ K10**); ao ser atingido o tempo de 2,0s, a Saída Y1 será acionada (**TA0 ≥ K20**); ao ser atingido o tempo de 3,0s, a Saída Y2 será acionada (**TA0 ≥ K30**); e ao ser atingido o tempo de 4,0s, será ativado o *Bit de Status* de T0 (Valor de *Preset* = K40). Na próxima execução do *rung* que contém o Temporizador T0, a

Entrada *Enable* não estará habilitada (*Bit de Status* de T0 ativado), reinicializando-o (**TA0 = 0**) e, conseqüentemente, desligando as Saídas Y0, Y1 e Y2. Como o Temporizador foi reinicializado, no *scan* seguinte, o *Bit de Status* de T0 não mais estará ativado (**TA0 = 0** e menor que o Valor de *Preset*), sendo iniciada nova temporização e repetindo indefinidamente a seqüência descrita.

A figura 4.48 apresenta o diagrama de tempo relativo à Lógica de Controle implementada na figura 4.47. Embora a Lógica de Controle determine o tempo de 1,0 segundo entre o acionamento das Saídas, o mesmo pode variar conforme o *Scan Time* do Programa de Aplicação.

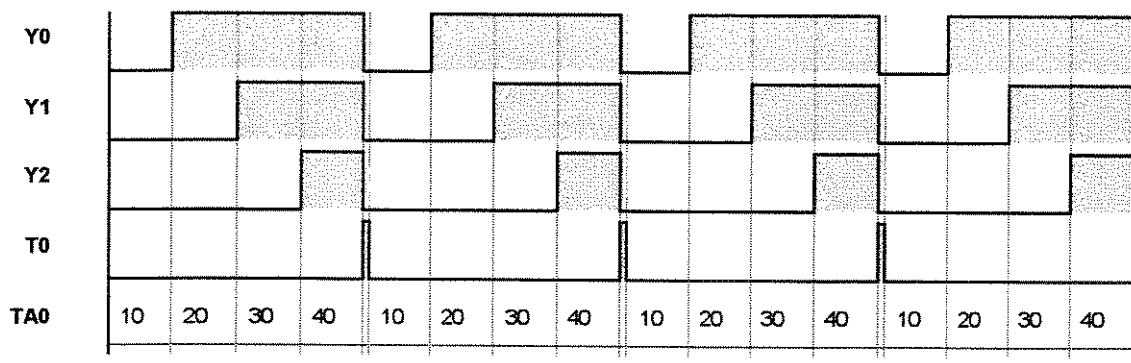


Figura 4.48 – Diagrama de Tempo (Temporizador Simples)

Temporizadores Acumuladores

• ***Accumulating Timer* (TMRA) / *Accumulating Fast Timer* (TMRAF)**

Os Temporizadores Acumuladores possuem duas Entradas de Controle – *Enable* (Habilitação) e *Reset* (Reinicialização). Ao ser acionada a Entrada *Enable* (*rung* = 1), e estando a Entrada *Reset* desligada (*rung* = 0), é iniciada a temporização. Ao ser desligada a Entrada *Enable* (*rung* = 0), o Valor Atual do Temporizador é mantido. Ao ser novamente acionada a Entrada *Enable*, o Valor Atual é incrementado, dando seqüência à temporização iniciada anteriormente. Ao ser acionada a Entrada *Reset* (*rung* = 1), independente da condição da Entrada *Enable* (*rung* = 0 ou *rung* = 1), o Temporizador é reinicializado (Valor Atual = 0), mantendo-se nesta condição até que a Entrada *Reset* seja desligada (*rung* = 0) e a Entrada *Enable* seja acionada (*rung* = 1), iniciando nova temporização. A figura 4.49 apresenta as especificações de um Temporizador Acumulador.

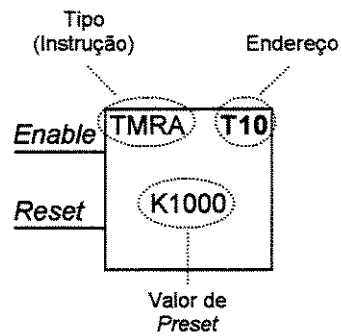


Figura 4.49 – Temporizador Acumulador

Especificações dos Temporizadores Acumuladores:

- *Tipo (Instrução)*: determina a base de tempo, podendo ser **TMRA** (incremento a cada décimo de segundo) ou **TMRAF** (incremento a cada centésimo de segundo – ***Fast***). Ambos operam com oito dígitos BCD (duas *Words* – 0 a 99999999), sendo que o Tipo TMRA temporiza de 0 a 9.999.999,9s e o Tipo TMRAF de 0 a 999.999,99s
- *Endereço*: em base octal, consome dois endereços consecutivos, e conseqüentemente dois Temporizadores do total disponível pela CPU. Por exemplo, se T0 for utilizado como Temporizador Acumulador (consumindo T0 e T1), o próximo Temporizador disponível será T2, independente deste ser utilizado como Temporizador Simples ou Acumulador. Possui um *Bit de Status* (*Bite* de Condição) – com mesmo endereço – relacionado, o qual é ativado quando o Valor Atual do Temporizador for igual ou superior ao Valor de *Preset* (Valor Pré-Configurado)
- *Valor de Preset*: determina o tempo que decorrerá, após a habilitação do Temporizador, para acionamento do *Bit de Status*. Os Operandos válidos como Valor de *Preset* são: **K** (Valor Constante), **V** (*Word*) e, em algumas CPUs, **P** (Ponteiro de Dados). Quando são utilizadas *Words* (Operando **V**) como Valores de *Preset*, o endereço indicado no Temporizador (por exemplo, V2000) contém os quatro dígitos menos significativos e o endereço seguinte (no caso, V2001), os quatro dígitos mais significativos. Esta consideração é válida também para Ponteiros de Dados (Operando **P**). O Valor de *Preset* deve ter formato BCD (0-99999999), sendo que valores iguais podem determinar tempos diferentes, conforme o Temporizador utilizado. Por exemplo, K10 determina 1,0 segundo

para Temporizadores TMRA (10 x 0,1 = 1,0) e 0,1 segundo para Temporizadores TMRAF (10 x 0,01 = 0,1)

- *Valor Atual*: embora não aparente na representação, o Valor Atual dos Temporizadores pode ser acessado através de *Words* específicas, sendo que V0 (ou TA0) contém os quatro dígitos menos significativos do Valor Atual de T0, e V1 (ou TA1) contém os quatro dígitos mais significativos do Valor Atual de T0, ou ainda os quatro dígitos menos significativos do Valor Atual de T1, e assim sucessivamente. Estas *Words* podem ser utilizadas na implementação da Lógica de Controle

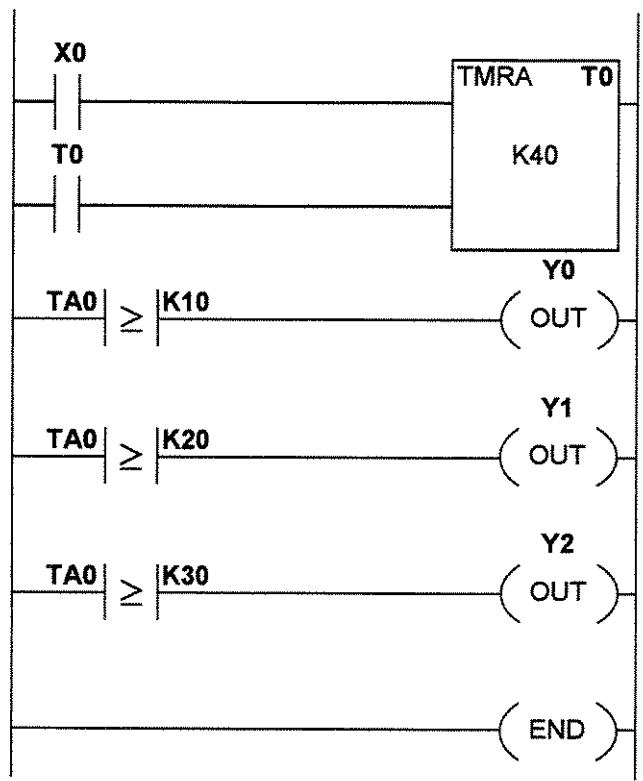


Figura 4.50 – Temporizador Acumulador, *Bit de Status* e Valor Atual

Uma vez acionada a Entrada *Enable* do Temporizador Acumulador (*rung* = 1), e estando a Entrada *Reset* desligada (*rung* = 0), o mesmo terá seu Valor Atual incrementado conforme a base de tempo (a cada 0,1s para TMRA, e a cada 0,01s para TMRAF) até que a mesma seja desligada (*rung* = 0), parando a temporizaçãqo e mantendo o Valor Atual do Temporizador. Ao ser novamente acionada a Entrada *Enable*, tem continuidade a temporização, partindo-se do Valor Acumulado. Se for atingido o valor máximo de temporização (9.999.999,9s para TMRA, e

999.999,99s para TMRAF), o Valor Atual do Temporizador não mais será incrementado (não retornando a zero automaticamente – *looping*), permanecendo neste valor até que o mesmo seja reinicializado. O acionamento da Entrada *Reset* (*rung* = 1) reinicializa o Temporizador Acumulador (Valor Atual = 0), tendo prioridade sobre a condição da Entrada *Enable*.

A figura 4.50 apresenta uma Lógica de Controle utilizando Temporizador Acumulador, e a figura 4.51 apresenta o diagrama de tempo relativo à esta implementação

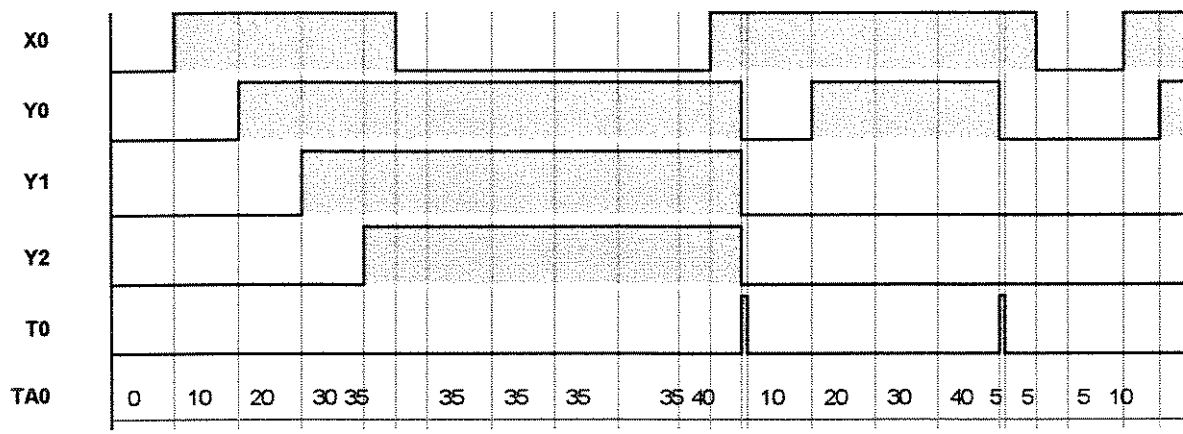


Figura 4.51 – Diagrama de Tempo (Temporizador Acumulador)

4.3.6 Contadores (*Counters*)

Normalmente são utilizados para contagem de condições e/ou eventos controlados pelo Programa de Aplicação. O modo de contagem (crescente/decrescente), o tipo e a quantidade de Contadores disponíveis, variam conforme o fabricante e/ou a Família de PLC.

As CPUs *PLC Direct by Koyo* dispõem de três tipos de Contadores: os Simples, os *Up/Down Counters* (Contadores Crescente/Decrescente) e os *Stage Counters* (Contador de Estágio). A quantidade depende da CPU utilizada (mínimo de 64 para as CPUs mais simples).

Os Contadores disponíveis podem ser utilizados como qualquer um dos tipos, conforme a necessidade da Aplicação, sendo que os *Up/Down Counters* consomem dois Contadores (dois endereços consecutivos – CT0 e CT1, por exemplo) do total disponível, por operarem com oito dígitos BCD (duas *Words*).

Contador Simples – *Counter* (CNT)

O Contador Simples possui duas Entradas de Controle – *Count* (Contagem) e *Reset* (Reinicialização). A cada transição de 0 para 1 da Entrada *Count* (*rung* = 0 → 1), e estando a Entrada *Reset* desligada (*rung* = 0), o Valor Atual do Contador é incrementado em uma unidade. Ao ser acionada a Entrada *Reset* (*rung* = 1), independente da condição da Entrada *Count* (*rung* = 0 ou *rung* = 1), o Contador é reinicializado (Valor Atual = 0), mantendo-se nesta condição até que a Entrada *Reset* seja desligada (*rung* = 0) e a Entrada *Count* seja novamente acionada (*rung* = 0 → 1), reiniciando a contagem. A figura 4.52 apresenta as especificações de um Contador Simples.

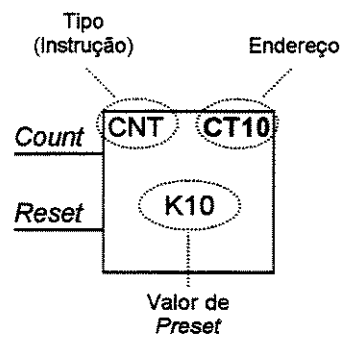


Figura 4.52 – Contador Simples

Especificações do Contador Simples:

- *Tipo (Instrução)*: determina Contador Simples (**CNT**), possuindo modo de contagem crescente apenas, e operando com quatro dígitos BCD (uma *Word* - 0 a 9999)
- *Endereço*: em base octal, possui um *Bit de Status* (*Bit* de Condição) – com mesmo endereço – relacionado, o qual é ativado quando o Valor Atual do Contador for igual ou superior ao Valor de *Preset* (Valor Pré-Configurado)
- *Valor de Preset*: determina a quantidade de transições de 0 para 1 da Entrada *Count*, com a Entrada *Reset* desligada (*rung* = 0), para acionamento do *Bit de Status*. Os Operandos válidos como Valor de *Preset* são: **K** (Valor Constante), **V** (*Word*) e, em algumas CPUs, **P** (Ponteiro de Dados). O Valor de *Preset* deve ter formato BCD (0-9999)
- *Valor Atual*: embora não aparente na representação, o Valor Atual do Contador pode ser acessado através de *Words* específicas, sendo que V1000 (ou CTA0) contém o Valor Atual

de CT0, V1001 (ou CTA1) contém o Valor Atual de CT1, e assim sucessivamente. Estas *Words* podem ser utilizadas na implementação da Lógica de Controle

A cada transição de 0 para 1 da Entrada *Count* (*rung* = 0 → 1), e estando a Entrada *Reset* desligada (*rung* = 0), o Valor Atual do Contador é incrementado em apenas uma unidade, independente do tempo que esta permaneça desligada (Entrada *Count* = 0) ou ligada (Entrada *Count* = 1). Porém, deve ser observado que as transições (0 → 1 e 1 → 0) devem ocorrer em *scans* distintos, devido à seqüência do Sistema de Operação da CPU (Leitura das Entradas → Execução do Programa de Aplicação → Escrita das Saídas). Se ocorrer mais de uma transição em um único *scan* (0 → 1 → 0 →...), o Valor Atual do Contador será incrementado em apenas uma unidade. Ao ser acionada a Entrada *Reset* (*rung* = 1), independente da condição da Entrada *Count* (*rung* = 0 ou *rung* = 1), o Contador é reinicializado (Valor Atual = 0), mantendo-se nesta condição até que a Entrada *Reset* seja desligada (*rung* = 0) e a Entrada *Count* seja novamente acionada (*rung* = 0 → 1), reiniciando a contagem. Ao ser atingido o valor máximo de contagem (9.999), o Valor Atual do Contador não mais será incrementado (não retornando a zero automaticamente - *looping*), permenecendo neste valor até que o mesmo seja reinicializado.

Ao iniciar a execução da Lógica de Controle implementada na figura 4.53, o *Bit de Status* de CT0 não estará ativado (**CTA0 = 0** e menor que o Valor de *Preset*), habilitando a contagem (Entrada *Reset* = 0). Assim, a cada transição de 0 para 1 (*off* → *on*) da Entrada X0, o Valor Atual do Contador CT0 será incrementado em uma unidade. Ao ser atingida a contagem de 1 transição, a Saída Y0 será acionada (**CTA0 ≥ K1**); ao ser atingida a contagem de 2 transições, a Saída Y1 será acionada (**CTA0 ≥ K2**); ao ser atingida a contagem de 3 transições, a Saída Y2 será acionada (**CTA0 ≥ K3**); e ao ser atingida a contagem de 4 transições, será ativado o *Bit de Status* de CT0 (Valor de Preset = K4). Na próxima execução do *rung* que contém o Contador CT0, a Entrada *Reset* estará acionada (*Bit de Status* de CT0 ativado), reinicializando-o (**CTA0 = 0**) e, conseqüentemente, desligando as Saídas Y0, Y1 e Y2. Como o Contador foi reinicializado, no *scan* seguinte, o *Bit de Status* de CT0 não mais estará ativado (**CTA0 = 0** e menor que o Valor de *Preset*), possibilitando o início de nova contagem na próxima transição de 0 para 1 (*off* → *on*) da Entrada X0.

A figura 4.54 apresenta o diagrama de tempo relativo à Lógica de Controle implementada na figura 4.53.

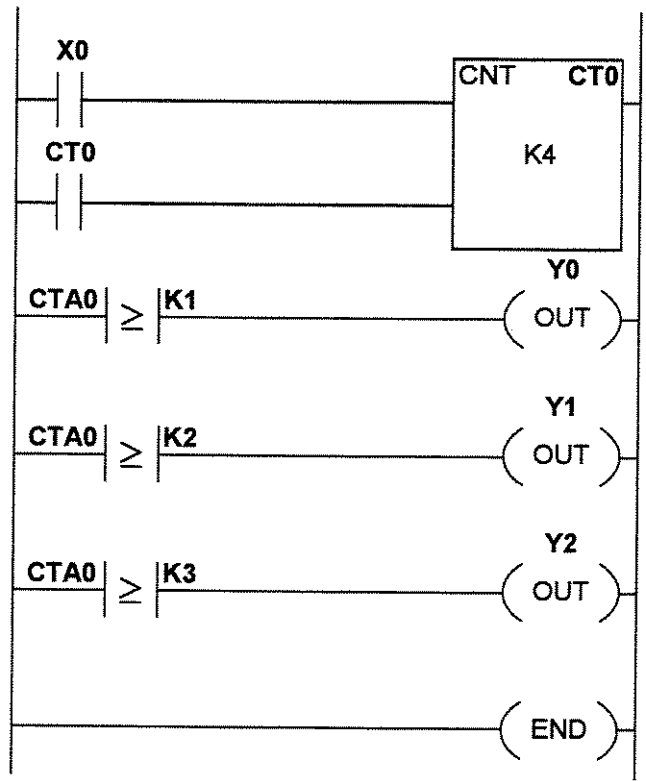


Figura 4.53 – Contador Simples, *Bit de Status* e Valor Atual

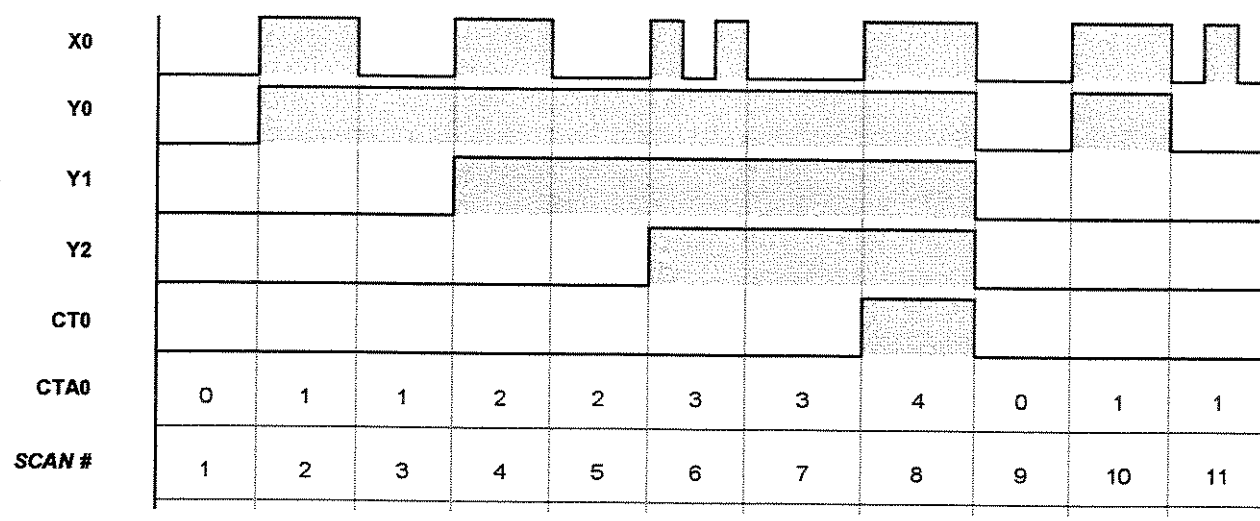


Figura 4.54 – Diagrama de Tempo (Contador Simples)

***Up/Down Counter* (UDC)**

O *Up/Down Counter* (Contador Crescente/Decrescente) possui três Entradas de Controle – *Up* (Contagem Crescente), *Down* (Contagem Decrescente) e *Reset* (Reinicialização). Estando a Entrada *Reset* desligada (*rung* = 0), a cada transição de 0 para 1 da Entrada *Up* (*rung* = 0 → 1), o Valor Atual do Contador é incrementado em uma unidade; e a cada transição de 0 para 1 da Entrada *Down* (*rung* = 0 → 1), o Valor Atual do Contador é decrementado em uma unidade. Ao ser acionada a Entrada *Reset* (*rung* = 1), independente da condição das Entradas *Up* e *Down* (*rungs* = 0 ou *rungs* = 1), o Contador é reinicializado (Valor Atual = 0), mantendo-se nesta condição até que a Entrada *Reset* seja desligada (*rung* = 0) e a Entrada *Up* seja novamente acionada (*rung* = 0 → 1), reiniciando a contagem. A figura 4.55 apresenta as especificações de um *Up/Down Counter*.

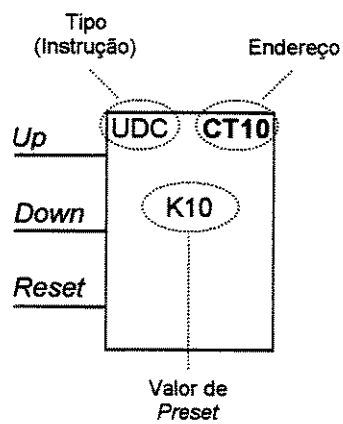


Figura 4.55 – *Up/Down Counter*

Especificações do *Up/Down Counter*:

- *Tipo (Instrução)*: determina *Up/Down Counter* (**UDC**), possuindo modo de contagem crescente/decrescente, e operando com oito dígitos BCD (duas *Words* - 0 a 99999999)
- *Endereço*: em base octal, consome dois endereços consecutivos, e conseqüentemente dois Contadores do total disponível pela CPU. Por exemplo, se CT0 for utilizado como *Up/Down Counter* (consumindo CT0 e CT1), o próximo Contador disponível será CT2, independente da utilização deste (Contador Simples, *Up/Down Counter* ou *Stage Counter*). Possui um *Bit de Status* (*Bit* de Condição) – com mesmo endereço – relacionado, o qual é

ativado quando o Valor Atual do Contador for igual ou superior ao Valor de *Preset* (Valor Pré-Configurado)

- *Valor de Preset*: determina a contagem a ser atingida, para acionamento do *Bit de Status*. Os Operandos válidos como Valor de *Preset* são: **K** (Valor Constante), **V** (*Word*) e, em algumas CPUs, **P** (Ponteiro de Dados). Quando são utilizadas *Words* (Operando **V**) como Valores de *Preset*, o endereço indicado no Contador (por exemplo, V2000) contém os quatro dígitos menos significativos e o endereço seguinte (no caso, V2001), os quatro dígitos mais significativos. Esta consideração é válida também para Ponteiros de Dados (Operando **P**). O Valor de *Preset* deve ter formato BCD (0-99999999)
- *Valor Atual*: embora não aparente na representação, o Valor Atual do Contador pode ser acessado através de *Words* específicas, sendo que V1000 (ou CTA0) contém os quatro dígitos menos significativos do Valor Atual de CT0, e V1001 (ou CTA1) contém os quatro dígitos mais significativos do Valor Atual de CT0, ou ainda os quatro dígitos menos significativos do Valor Atual de CT1, e assim sucessivamente. Estas *Words* podem ser utilizadas na implementação da Lógica de Controle

Estando as Entradas *Down* e *Reset* desligadas (*rungs* = 0), a cada transição de 0 para 1 da Entrada *Up* (*rung* = 0 → 1), o Valor Atual do Contador é incrementado em apenas uma unidade, independente do tempo que esta permaneça desligada (Entrada *Up* = 0) ou ligada (Entrada *Up* = 1); e estando as Entradas *Up* e *Reset* desligadas (*rungs* = 0), a cada transição de 0 para 1 da Entrada *Down* (*rung* = 0 → 1), o Valor Atual do Contador é decrementado em apenas uma unidade, independente do tempo que esta permaneça desligada (Entrada *Down* = 0) ou ligada (Entrada *Down* = 1). Porém, deve ser observado que as transições (0 → 1 e 1 → 0) das Entradas *Up* e *Down* devem ocorrer em *scans* distintos e não simultâneos, devido à seqüência do Sistema de Operação da CPU (Leitura das Entradas → Execução do Programa de Aplicação → Escrita das Saídas). Se ocorrer mais de uma transição em um único *scan* (0 → 1 → 0 →...) das Entradas *Up* ou *Down*, o Valor Atual do Contador será incrementado ou decrementado em apenas uma unidade. Se ocorrerem transições simultâneas das Entradas *Up* (*rung* = 0 → 1) e *Down* (*rung* = 0 → 1) em um mesmo *scan*, o Valor Atual do Contador não será alterado, sendo desconsideradas tais transições. Ao ser acionada a Entrada *Reset* (*rung* = 1), independente da condição das Entradas *Up* e *Down* (*rungs* = 0 ou *rungs* = 1), o Contador é reinicializado (Valor Atual = 0),

mantendo-se nesta condição até que a Entrada *Reset* seja desligada (*rung* = 0) e a Entrada *Up* seja novamente acionada (*rung* = 0 → 1), reiniciando a contagem.

Ao ser atingido o Valor de *Preset*, o *Bit de Status* do Contador é acionado, permanecendo nesta condição enquanto o Valor Atual do Contador for igual ou superior ao Valor de *Preset*. No entanto, se o *Bit de Status* estiver acionado (Valor Atual ≥ Valor de *Preset*) e ocorrerem transições na Entrada *Down* fazendo com que o Valor Atual se torne menor que o Valor de *Preset*, o *Bit de Status* será desligado, sendo acionado novamente quando o Valor Atual for igual ou superior ao Valor de *Preset*. Ao ser atingido o valor máximo de contagem (99.999.999), o Valor Atual do Contador não mais será incrementado (não retornando a zero automaticamente – *looping*), permenecendo neste valor até que o mesmo seja reinicializado ou até que a Entrada *Down* seja acionada, decrementando tal valor. Ao ser atingido o valor mínimo de contagem (0) – por transições da Entrada *Down* ou por acionamento da Entrada *Reset*, o Valor Atual do Contador não será mais decrementado (não permitindo contagem negativa: -99.999.999, -99.999.998,...), permenecendo neste valor até que a Entrada *Up* seja novamente acionada (*rung* = 0 → 1), reiniciando contagem.

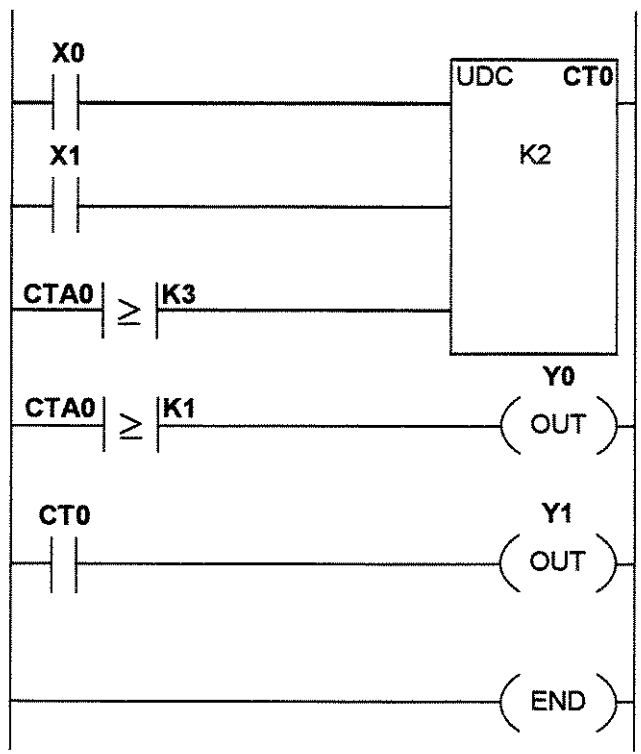


Figura 4.56 – *Up/Down Counter, Bit de Status* e Valor Atual

A figura 4.56 apresenta uma Lógica de Controle utilizando *Up/Down Counter* e a figura 4.57 apresenta o diagrama de tempo relativo a esta implementação.

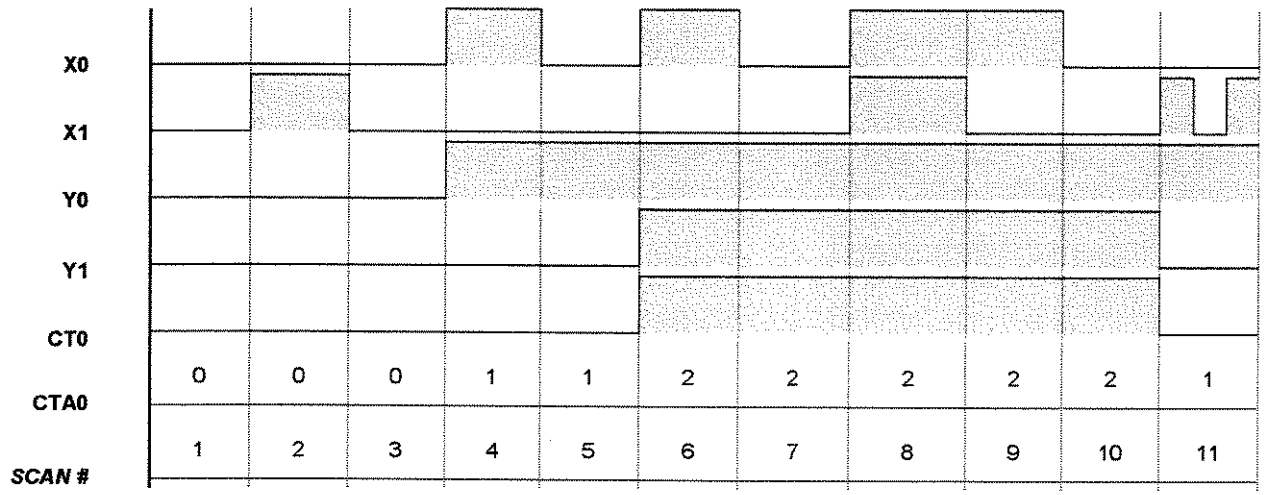


Figura 4.57 – Diagrama de Tempo (*Up/Down Counter*)

Stage Counter (SGCNT)

O *Stage Counter* (Contador de Estágio) possui apenas uma Entrada de Controle – *Count* (Contagem), que a cada transição de 0 para 1 (*rung* = 0 → 1) incrementa uma unidade ao Valor Atual do Contador. A reinicialização (Valor Atual = 0) é feita através da Instrução *Reset* (RST) referenciada ao endereço do Contador utilizado. A principal utilização deste Tipo de Contador, porém não restrita, é na Programação por Estágios (Instruções RLL^{PLUS}). A figura 4.58 apresenta as especificações do *Stage Counter*.

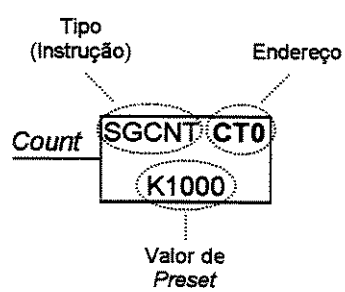


Figura 4.58 – *Stage Counter*

Especificações do *Stage Counter*:

- *Tipo (Instrução)*: determina *Stage Counter* (**SGCNT**), possuindo modo de contagem apenas crescente, sendo reinicializado pela Instrução *Reset*, e operando com quatro dígitos BCD (uma *Word* - 0 a 9999)
- *Endereço*: em base octal, possui um *Bit de Status* (*Bit* de Condição) – com mesmo endereço – relacionado, o qual é ativado quando o Valor Atual do Contador for igual ou superior ao Valor de *Preset* (Valor Pré-Configurado)
- *Valor de Preset*: determina a quantidade de transições de 0 para 1 da Entrada *Count*, sem execução da Instrução *Reset* referenciada ao Contador utilizado, para acionamento do *Bit de Status*. Os Operandos válidos como Valor de *Preset* são: **K** (Valor Constante), **V** (*Word*) e, em algumas CPUs, **P** (Ponteiro de Dados). O Valor de *Preset* deve ter formato BCD (0-9999)
- *Valor Atual*: embora não aparente na representação, o Valor Atual do Contador pode ser acessado através de *Words* específicas, sendo que V1000 (ou CTA0) contém o Valor Atual de CT0, V1001 (ou CTA1) contém o Valor Atual de CT1, e assim sucessivamente. Estas *Words* podem ser utilizadas na implementação da Lógica de Controle

A cada transição de 0 para 1 da Entrada *Count* (*rung* = 0 → 1), o Valor Atual do Contador é incrementado em apenas uma unidade, independente do tempo que esta permaneça desligada (Entrada *Count* = 0) ou ligada (Entrada *Count* = 1). Porém, deve ser observado que as transições (0 → 1 e 1 → 0) devem ocorrer em *scans* distintos, devido à seqüência do Sistema de Operação da CPU (Leitura das Entradas → Execução do Programa de Aplicação → Escrita das Saídas). Se ocorrer mais de uma transição em um único *scan* (0 → 1 → 0 →...), o Valor Atual do Contador será incrementado em apenas uma unidade. Ao ser executada a Instrução *Reset* (*rung* = 1) referenciada ao *Stage Counter* utilizado, independente da condição da Entrada *Count* (*rung* = 0 ou *rung* = 1), o Contador é reinicializado (Valor Atual = 0), mantendo-se nesta condição até que a Instrução *Reset* não seja mais executada (*rung* = 0) e a Entrada *Count* seja novamente acionada (*rung* = 0 → 1), reiniciando a contagem. Ao ser atingido o valor máximo de contagem (9.999), o Valor Atual do Contador não mais será incrementado (não retornando a zero automaticamente – *looping*), permanecendo neste valor até que o mesmo seja reinicializado.

A figura 4.59 apresenta uma Lógica de Controle utilizando *Stage Counter* e a figura 4.60 apresenta o diagrama de tempo relativo a esta implementação.

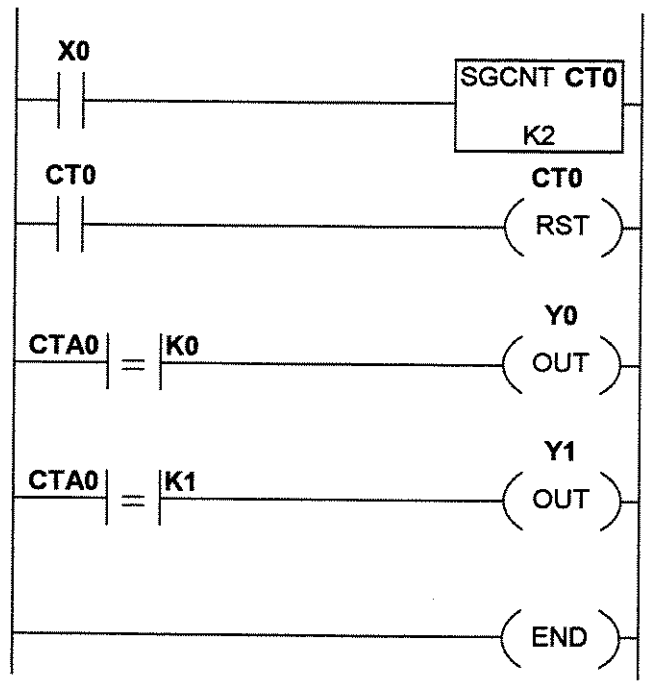


Figura 4.59 – *Stage Counter*, *Bit de Status* e Valor Atual

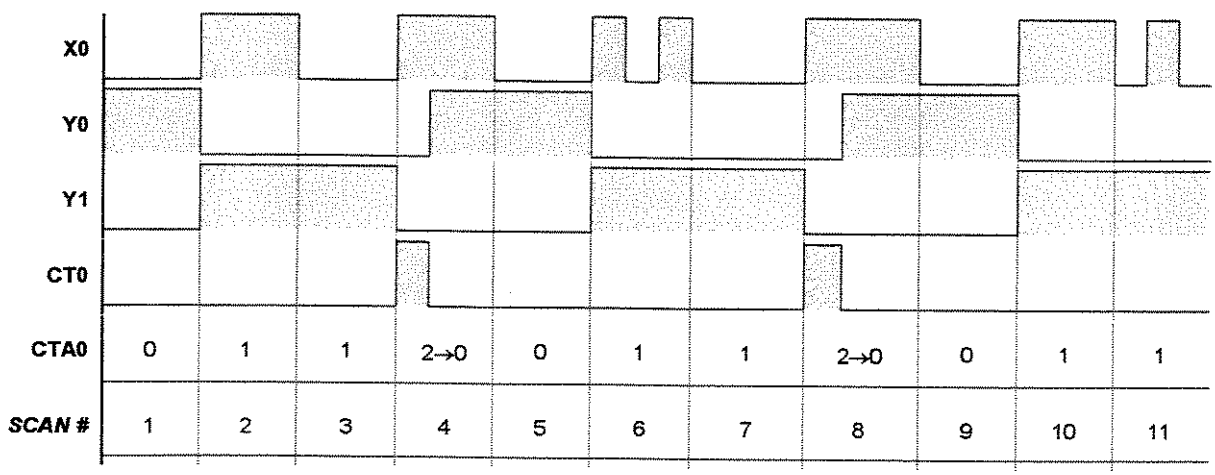


Figura 4.60 – Diagrama de Tempo (*Stage Counter*)

4.4 Programação por Estágios (Instruções RLL^{PLUS} – *Relay Ladder Logic Plus*)

A Programação por Estágios – através de Instruções RLL^{PLUS}, disponível praticamente em todas as CPUs *PLC Direct by Koyo* (exceto em algumas CPUs da Família DL305), proporciona o

desenvolvimento de Programas de Aplicação estruturados, permitindo a 'tradução' da descrição do Sistema Automatizado – elaborada em *SFC* (*Sequential Function Chart* – Diagrama Funcional Seqüencial), para Linguagem Ladder. É classificada como Método Avançado de Programação, sendo que as Instruções RLL^{PLUS} não são consideradas padrões (disponíveis na maioria dos PLCs).

As principais desvantagens apresentadas por Programas de Aplicação desenvolvidos em Linguagem Ladder, utilizando apenas Métodos Convencionais de Programação, são:

- necessidade de criação de múltiplos inter-travamentos para verificação constante de condições
- dificuldade de localização do problema durante depuração/manutenção, pois 'todos' os *rungs* devem ser verificados
- dificuldade de alterações posteriores nos programas, devido à complexidade da Lógica de Controle implementada
- dificuldade de gerenciamento de programas maiores, devido à falta de estruturação
- maior tempo de execução (*scan time*), pois todos os *rungs* devem ser executados pela CPU (mesmo aqueles que não interfiram no controle em dado momento)

Normalmente, a documentação referente ao Projeto também representa uma dificuldade no desenvolvimento de Programas de Aplicação, utilizando apenas Métodos Convencionais. Muitas vezes, a implementação da Lógica de Controle acaba se tornando uma seqüência de *tentativas* → *erros* → *correções*, quase infindável, por falta ou ineficiência de documentação.

Estas desvantagens podem ser facilmente eliminadas através da Programação Por Estágios, como apresentado a seguir.

4.4.1 Vantagens da 'Programação por Estágios'

Qualquer Sistema Automatizado (Máquina ou Processo Industriais, por exemplo), independente da complexidade, pode ser descrito por uma combinação de etapas (operações) seqüenciais e/ou paralelos. O Programa de Aplicação completo pode parecer complexo, mas cada etapa pode ser extremamente simples. Normalmente, quanto maior a divisão do Programa de

Aplicação, maior a simplicidade das etapas. As Instruções RLL^{PLUS} permitem a 'quebra' (divisão) do Programa de Aplicação em tarefas simples, chamadas de Estágios. Cada Estágio representa uma etapa, que corresponde diretamente à Seqüência de Operação do Sistema Automatizado.

Como a Programação por Estágios permite a 'tradução', de forma simples e direta, da descrição do Sistema Automatizado – elaborada em *SFC* (*Sequential Function Chart*) ou em Fluxograma (*Flowchart*), para a Linguagem Ladder, conclui-se que se os mesmos atenderem corretamente todas as necessidades da Aplicação, a Lógica de Controle implementada também atenderá. Esta característica proporciona:

- facilidade de gerenciamento do Programa de Aplicação, independente de tamanho e/ou complexidade, devido à estruturação do mesmo
- facilidade de alterações posteriores, devido à clareza da documentação utilizada para implementação da Lógica de Controle
- diminuição do tempo de depuração/manutenção, devido à facilidade de localização e entendimento do problema

Porém, a maior vantagem da Programação por Estágios está na forma como a CPU executa o Programa de Aplicação. Durante a execução de Programas desenvolvidos com Métodos Convencionais, os *rungs* são executados seguida e indistintamente, ou seja, a maior parte do tempo é consumida ('gasta') verificando condições desnecessárias àquela Seqüência de Operação – inter-travamentos, proteções, etc. Na execução de Programas estruturados por Estágios (Instruções RLL^{PLUS}), a CPU executa apenas a Lógica de Controle presente no(s) Estágio(s) ativo(s), ou seja, consome-se tempo verificando apenas condições necessárias à Seqüência de Operação naquele momento.

4.4.2 Conceitos Básicos da 'Programação por Estágios'

Os conceitos apresentados a seguir, baseados nas Instruções RLL^{PLUS} disponíveis nas CPUs *PLC Direct by Koyo*, devem ser observados durante o desenvolvimento de Programas de Aplicação utilizando Estágios.

Endereçamento dos Estágios

Conforme apresentado, quanto maior a quantidade de Estágios, maior a simplicidade da Lógica de Controle executada em cada Estágio. Porém, a quantidade máxima de Estágios que podem ser criados depende da CPU utilizada (mínimo de 256 para as CPUs mais simples).

Os Estágios são identificados por 'S' e endereçados em base octal (S0,..., S7, S10,...). Não há necessidade de se utilizar os endereços seqüencialmente (S0–S1–S2–S3–...), podendo haver endereços livres entre os Estágios utilizados (S0–S4–S10–S14–...). Também, não há necessidade de estarem localizados seqüencialmente no Programa de Aplicação (endereços em ordem crescente), podendo ser criada a seqüência que for mais conveniente ao programador (S20→S34→S0→S127→S5→...). Cada Estágio possui um *Bit de Status* (*Bit* de Condição) – com mesmo endereço – que pode ser associado às Instruções Booleanas de Entrada (indicando se o Estágio está ativo ou não) ou de Saída (para ativar o Estágio).

As restrições que devem ser observadas são: a quantidade máxima de Estágios disponíveis, e a utilização de cada endereço em um único Estágio (não podem existir dois ou mais Estágios S0, por exemplo).

Tipos de Estágios

Existem basicamente dois Tipos de Estágios: os Iniciais (*Initial Stages* – ISG), que são ativados automaticamente no primeiro *scan* da CPU (*Power-Up*); e os Comuns (*Stages* – SG), que são ativados através de Instruções executadas pela Lógica de Controle. Podem existir tantos Estágios Iniciais (ISG) quantos se fizerem necessários, sendo que todos serão ativados automaticamente durante o primeiro *scan* da CPU. A figura 4.61 apresenta estes dois Tipos de Estágios.

Controle dos Estágios

Um Estágio pode ser ativado:

- automaticamente no *Power-Up*, através da utilização de Estágios Iniciais (ISG)
- através de Instruções *Jump* (JMP), ativando o Estágio referenciado na Instrução e desativando o Estágio que a executou

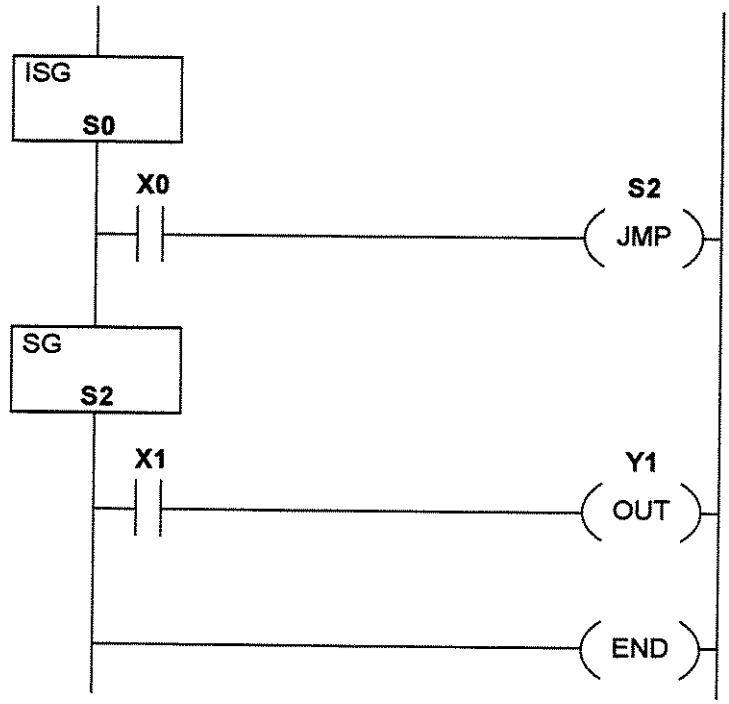


Figura 4.61 – Estágio Inicial (ISG) e Estágio Comum (SG)

- através de Instruções *Set* (SET), ativando o Estágio referenciado na Instrução e mantendo ativado o Estágio que a executou
- através de transição direta (*rung*), sem necessidade de Instrução. Este recurso pode ser utilizado para ativar apenas o Estágio imediatamente seguinte

E pode ser desativado:

- automaticamente no *Power-Up*, através da utilização de Estágios Comuns (SG)
- através de Instruções *Jump* (JMP), desativando o Estágio que executou a Instrução e ativando o Estágio referenciado na mesma
- através de Instruções *Reset* (RST), desativando o Estágio referenciado na Instrução e mantendo ativado o Estágio que a executou

A figura 4.62 apresenta a ativação direta de um Estágio.

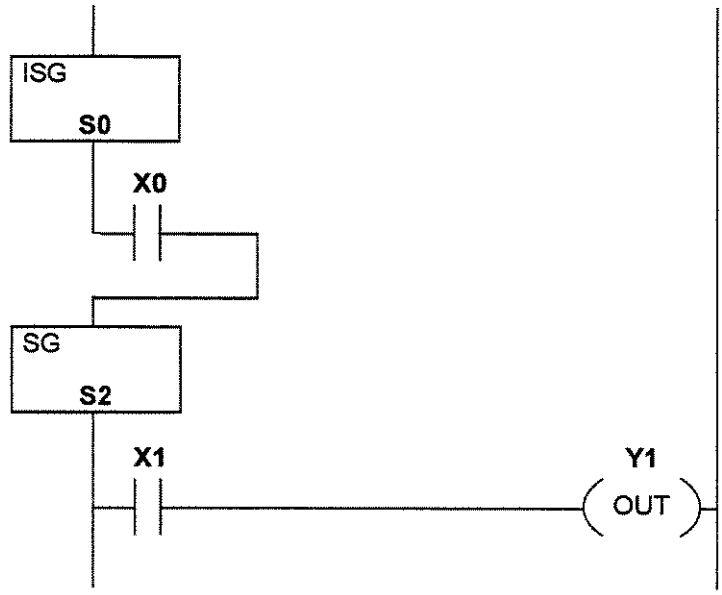


Figura 4.62 – Ativação de Estágio por Transição Direta

Execução da Lógica de Controle

Algumas considerações devem feitas quanto à execução das Instruções presentes na Lógica de Controle implementada em cada Estágio do Programa de Aplicação:

- ***Instrução 'Out'***
  1. Se a Instrução estiver sendo executada (Saída referenciada acionada) e o Estágio for desativado, a mesma deixa de ser executada (desligando a Saída referenciada)
  2. Múltiplas Instruções podem ser referenciadas ao mesmo Ponto de Saída, desde que em Estágios diferentes e que apenas um esteja ativo em cada *scan*. Caso contrário, terá controle real sobre o Ponto de Saída, a última condição a ser executada
- ***Instruções 'Set'e 'Reset'***
  1. A desativação do Estágio não interfere na Saída referenciada (mantém-se a condição anterior), necessitando de nova execução das Instruções para alteração da mesma
- ***Instrução 'Positive Differential'***
  1. Se o Estágio for ativado e o *rung* já estiver acionado (*rung* = 1), a Instrução será executada, embora não tenha ocorrido a transição de 0 para 1 do *rung* que a controla (*rung* = 0 → 1) durante o *scan* de ativação do Estágio

- ***Temporizadores***
  1. A desativação do Estágio faz com que os Temporizadores Simples (TMR ou TMRF) sejam reinicializados (Valor Atual = 0), mantendo-os nesta condição até que o Estágio seja novamente ativado e a Entrada *Enable* acionada
  2. A desativação do Estágio não interfere nos Temporizadores Acumuladores (TMRA ou TMRAF), sendo mantido o Valor Atual dos mesmos até que o Estágio seja novamente ativado e a Entrada *Enable* acionada
  3. Os Temporizadores Acumuladores (TMRA ou TMRAF) podem ser reinicializados (Valor Atual = 0) através de Instrução *Reset* executada a partir de qualquer Estágio ativo, sendo referenciada aos dois endereços consumidos (*'Reset T0 T1'*, por exemplo)
- ***Contadores***
  1. Se o Estágio for ativado e as Entradas de Contagem (*Count*, *Up* ou *Down*) estiverem acionadas (*rung* = 1), o Valor Atual do Contador não será alterado, necessitando nova transição de 0 para 1 (*rung* = 0 → 1) das mesmas
  2. A desativação do Estágio não interfere nos Contadores (CNT, UDC ou SGCNT), sendo mantido o Valor Atual dos mesmos até que o Estágio seja novamente ativado e as Entradas *Count*, *Up* ou *Down* sejam acionadas, conforme o caso
  3. Os Contadores podem ser reinicializados (Valor Atual = 0) através de Instrução *Reset* (*'Reset CT0'*, por exemplo) executada a partir de qualquer Estágio ativo. Para os *Up/Down Counters*, faz-se necessário referenciar a Instrução aos dois endereços consumidos (*'Reset CT0 CT1'*, por exemplo)

Acionamento Incondicional

Para executar determinada Instrução de Saída (Booleana ou Temporizador Simples, por exemplo) incondicionalmente, pode-se posicionar o *rung* de controle logo após o início do Estágio (ISG ou SG), sem utilizar Relés Especiais (SP1, por exemplo) ou Relés de Controle exclusivamente utilizados para este fim.

Na Lógica de Controle implementada na figura 4.63, ao ser ativado o Estágio S2 (pelo acionamento de X0, que ativa S2 e desativa S0), a Saída.Y1 será acionada incondicionalmente (**Y1 = 1**) e a Saída Y2 será acionada a partir da condição de X1 (**X1 = 1**).

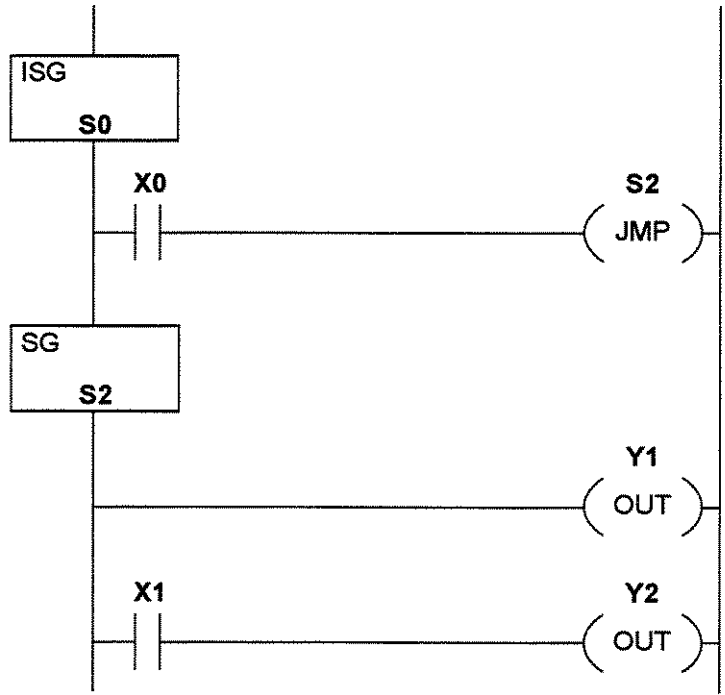


Figura 4.63 – Acionamento Incondicional

4.4.3 Instruções RLL^{PLUS}

A seguir, são apresentadas as Instruções RLL^{PLUS} atualmente disponíveis, na maioria das CPUs *PLC Direct by Koyo*, para implementação da Lógica de Controle utilizando a Programação por Estágios.

Initial Stage (ISG) e *Stage* (SG)

As Instruções *Initial Stage* e *Stage*, apresentadas na figura 4.64, indicam o início de um Estágio, e compreendem toda a Lógica de Controle contida até o início do Estágio seguinte.

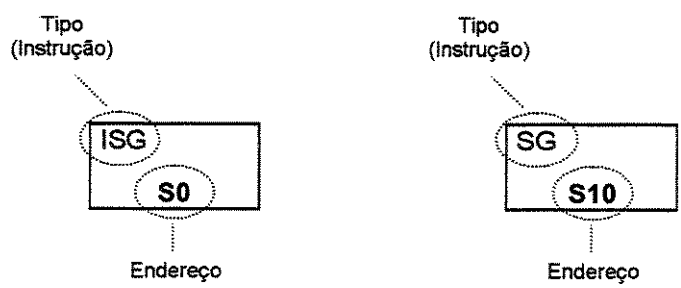


Figura 4.64 – *Initial Stage* (ISG) e *Stage* (SG)

A Instrução *Initial Stage* (ISG) define um Estágio que será ativado automaticamente no primeiro *scan* da CPU, permanecendo nesta condição até ser desativado pela execução das Instruções *Jump* ou *Reset*, ou por transição direta ao Estágio seguinte. Uma vez desativado, o *Initial Stage* passa a ser tratado como um Estágio Comum, podendo ser ativado novamente através de Instruções executadas pela Lógica de Controle. Um Programa de Aplicação pode ter múltiplos Estágios designados como ISG.

A Instrução *Stage* (SG) define um Estágio que estará desativado no primeiro *scan* da CPU, permanecendo nesta condição até ser ativado pela execução das Instruções *Jump* ou *Set*, ou por transição direta do Estágio anterior. Uma vez ativado, o *Stage* pode ser desativado através de Instruções executadas pela Lógica de Controle.

Se a Instrução de ativação/desativação (*Jump*, *Set* ou *Reset*) de determinado Estágio (ISG ou SG) estiver localizada antes do início do mesmo, terá efeito no próprio *scan* em que for executada. Caso contrário, se estiver localizada após o início do Estágio referenciado, terá efeito no *scan* imediatamente seguinte. Isto se deve à seqüência de execução do Programa de Aplicação (da esquerda para direita, e de cima para baixo, até encontrar a Instrução END).

***Jump* (JMP) e *Not Jump* (NJMP)**

As Instruções *Jump* e *Not Jump*, apresentadas na figura 4.65, possibilitam realizar a transição de um Estágio ativo, que as executa, para o Estágio referenciado nas mesmas.

Figura 4.65 – *Jump* (JMP) e *Not Jump* (JMP)

A Instrução *Jump* será executada se o *rung* que a controla estiver acionado (*rung* = 1), ativando o Estágio referenciado e desativando o Estágio que a executou. Caso contrário (*rung* = 0), a Instrução não será executa.

A Instrução *Not Jump* será executada se o *rung* que a controla não estiver acionado (*rung* = 0), ativando o Estágio referenciado e desativando o Estágio que a executou. Caso contrário (*rung* = 1), a Instrução não será executa.

Apesar das Instruções serem *Jump* (salto, pulo) e *Not Jump*, em condição alguma ocorre o desvio (salto) para o Estágio referenciado. A Lógica de Controle, existente entre o *rung* da Instrução e o Estágio referenciado, é executada normalmente. Estas Instruções apenas ativam o Estágio, ou seja, habilitam – ou não – a execução da Lógica de Controle contida no mesmo.

Se o Estágio referenciado nas Instruções *Jump* e *Not Jump* estiver localizado após o *rung* que as controla, será ativado no mesmo *scan* em que as Instruções forem executadas. Caso contrário, se estiver localizado antes do *rung* que as controla, será ativado no *scan* imediatamente seguinte. O Estágio que executa as Instruções *Jump* e *Not Jump* estará desativado no *scan* seguinte à execução das mesmas. No entanto, a Lógica de Controle contida neste Estágio e localizada após o *rung* que controla as Instruções, será executada ainda no *scan* de execução das mesmas.

Além das Instruções *Jump* e *Not Jump*, as Instruções Booleanas de Saída *Set* e *Reset* podem ser utilizadas para ativação e desativação de um Estágio, respectivamente. Para isto, basta referenciá-las ao Estágio (ou à faixa de Estágios) que se deseja ativar ou desativar. A diferença entre utilizar as Instruções *Jump* e *Not Jump* ou as Instruções *Set* e *Reset*, está no fato de que estas últimas não desativam o Estágio que as executa.

Ao iniciar a execução da Lógica de Controle implementada na figura 4.66, apenas o Estágio S0 estará ativo (ISG). Assim, ao ser atuada a Entrada X0, a Saída Y0 será acionada; e ao ser atuada a Entrada X1, serão ativados os Estágios S2 e S4 simultaneamente, desativando o Estágio S0 (Instruções *Jump*). A Lógica de Controle do Estágio S2 determina que ao ser atuada a Entrada X4, será acionada a Saída Y1; e ao ser atuada a Entrada X5, será desativado o próprio Estágio S2 (Instrução *Reset*). E a Lógica de Controle do Estágio S4 determina que ao ser atuada a Entrada X6, será atuada a Saída Y2; e ao ser desativado o Estágio S2 (através da atuação de X5), o

Estágio S0 será ativado novamente, desativando o Estágio S4 (Instrução *Jump*), retornando à condição inicial, ou seja, apenas o Estágio S0 ativo.

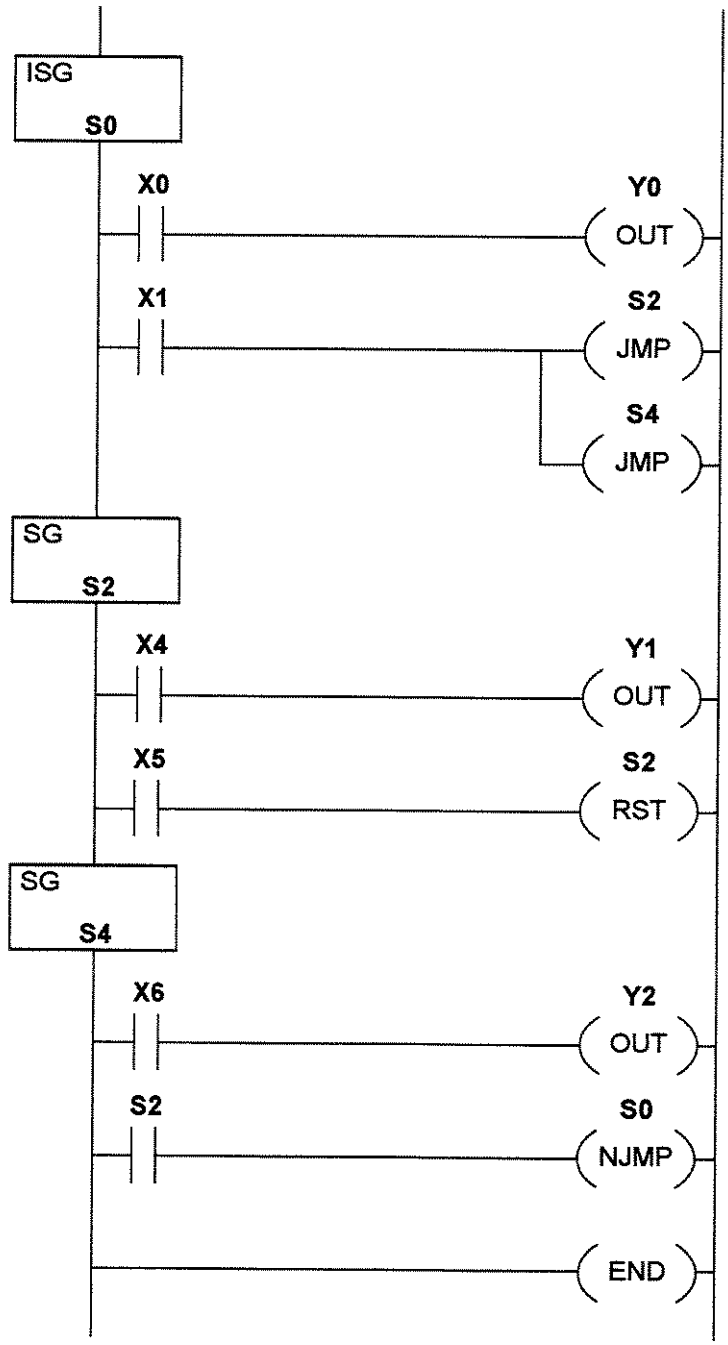


Figura 4.66 – Utilização de *Jump* (JMP) e *Not Jump* (NJMP)

***Converge Stage* (CV) e *Converge Jump* (CVJMP)**

Quando se deseja que condições contidas em Estágios distintos sejam responsáveis pela ativação de um determinado Estágio (ativação convergente), utilizam-se as Instruções *Converge Stage* e *Converge Jump*, apresentadas na figura 4.67.

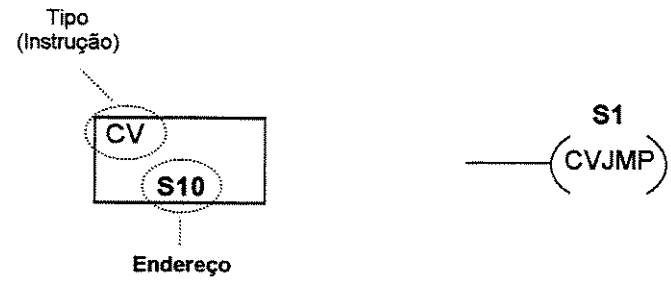


Figura 4.67 – *Converge Stage* (CV) e *Converge Jump* (CVJMP)

A figura 4.68 exemplifica a utilização das Instruções *Converge Stage* e *Converge Jump*, na qual os Estágios S2 e S4 são do Tipo CV, convergindo para o Estágio S5 (Tipo SG) que contém a Lógica de Controle a ser executada.

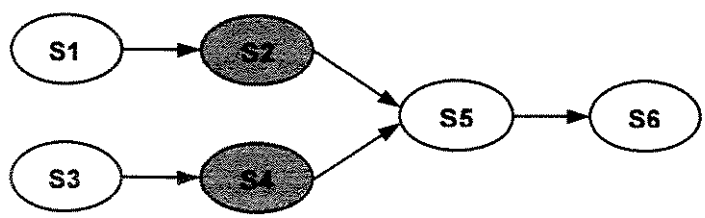


Figura 4.68 – Exemplo de Estágios Convergentes (S2 e S4)

Os Estágios CVs são 'Estágios Intermediários' apenas e devem ser dispostos de tal forma no Programa de Aplicação que não haja qualquer Lógica de Controle entre eles, sendo localizados seqüencialmente – formando um 'Grupo de Estágios'. O último Estágio CV ativa o Estágio controlado através da Instrução *Converge Jump*, desde que todos os demais Estágios do Grupo estejam ativados.

Apenas entre o último Estágio Tipo CV, de um determinado Grupo, e o Estágio ativado pela Instrução *Converge Jump* é permitido qualquer Lógica de Controle, a qual será executada enquanto todos os Estágios agrupados anteriormente estiverem devidamente ativados.

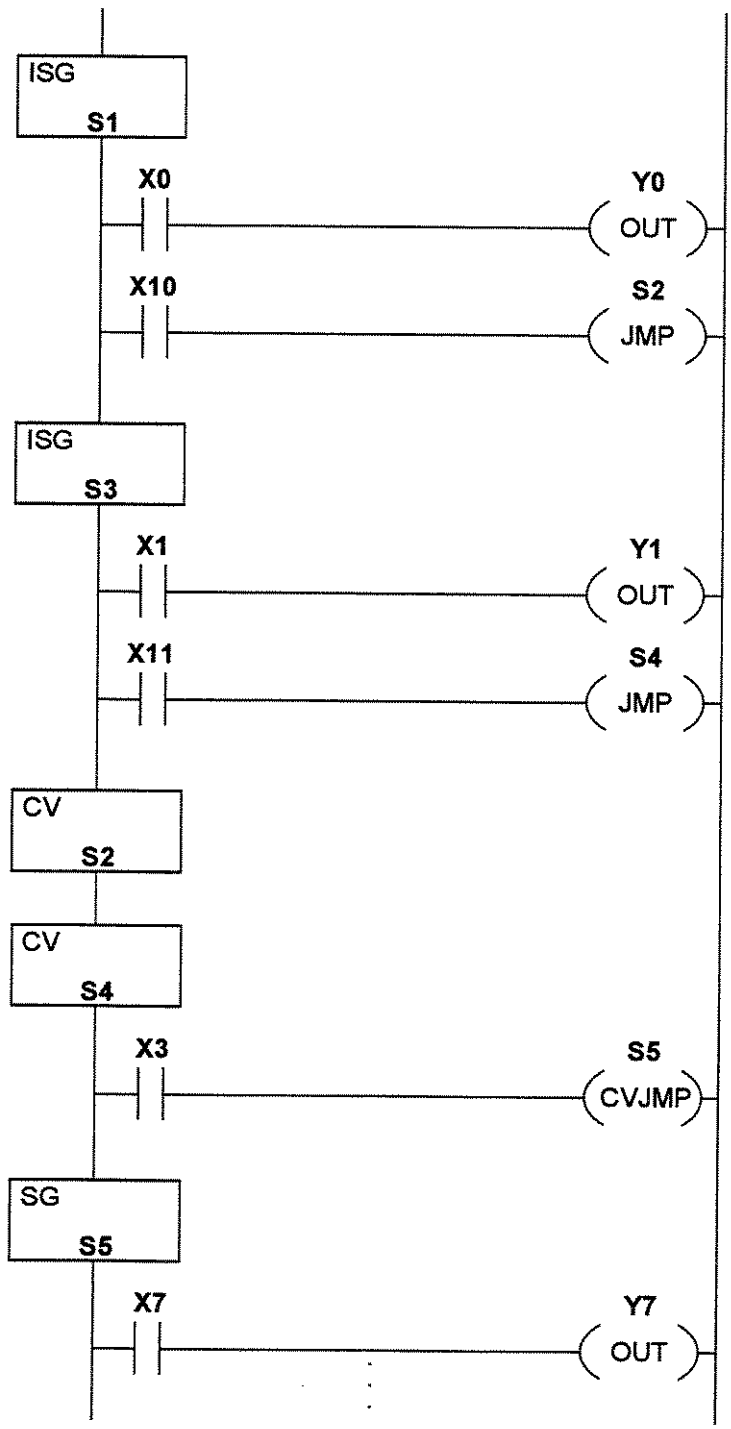


Figura 4.69 – Utilização de *Converge Stage* (CV) e *Converge Jump* (CVJMP)

Os Estágios Tipo CV podem ser ativados por Instruções *Jump*, *Not Jump* ou *Set*, sendo desativados simultaneamente pela execução da Instrução *Converge Jump* relativa ao Grupo de Estágios a que pertencem.

Na Lógica de Controle implementada na figura 4.69, o Estágio S5 apenas será executado se os Estágios S2 e S4 (Tipo CV) forem ativados (através de Instruções *Jump*, no caso) e a Entrada X3 for atuada, ativando o Estágio S5 e desativando os Estágios S2 e S4.

***Block Call* (BCALL), *Block* (BLK) e *Block End* (BEND)**

As Instruções *Block Call*, *Block* e *Block End*, apresentadas na figura 4.70, têm como finalidade facilitar a organização e o gerenciamento do Programa de Aplicação, através do agrupamento de Estágios em Blocos distintos.

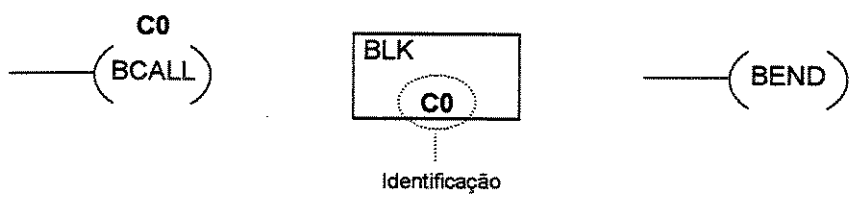


Figura 4.70 – *Block Call* (BCALL), *Block* (BLK) e *Block End* (BEND)

A Instrução *Block* determina o início de um Bloco de Estágios e a Instrução *Block End* (incondicional) o final do mesmo, sendo ativado ('chamado') através da Instrução *Block Call*. Cada Bloco de Estágios é identificado por um Relé de Controle (Dado Tipo C – C0, C1,...) único, utilizado apenas para este fim – não podendo ser utilizado em Instruções de Saída (*Out*, *Set*, *Reset*,...). Não é permitido ter dois ou mais Blocos identificados com o mesmo Relé de Controle.

Para que os Estágios de determinado Bloco sejam executados pela Lógica de Controle, faz-se necessário que o mesmo seja ativado através da execução constante da Instrução *Block Call*, podendo haver mais de um Bloco ativo ao mesmo tempo. A não execução desta Instrução desativa automaticamente todos os Estágios contidos no Bloco referenciado.

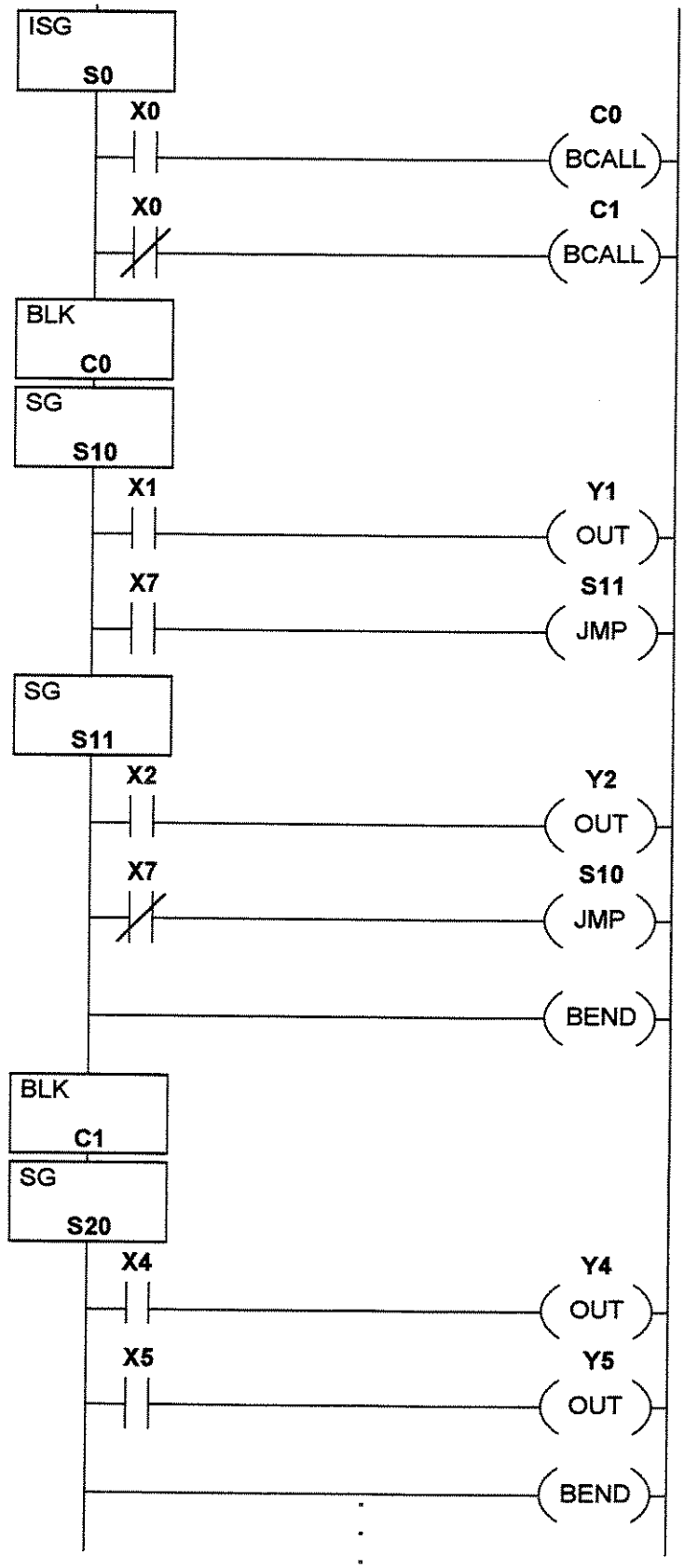


Figura 4.71 – Utilização de *Block Call* (BCALL), *Block* (BLK) e *Block End* (BEND)

Podem haver tantos Blocos de Estágios quantos se fizerem necessários, sendo o único limite a quantidade de Relés de Controle disponível em cada CPU.

Na composição dos Blocos não é possível a existência de Estágios Iniciais (ISG), sendo que imediatamente após cada Instrução *Block* deve iniciar-se um Estágio Comum (SG), o qual será automaticamente ativado na primeira execução da Instrução *Block Call* correspondente.

Quando se opta pelo agrupamento de Estágios em Blocos, não se faz necessário que todos os Estágios pertençam a algum Bloco, podendo haver Estágios independentes.

Na Lógica de Controle implementada na figura 4.71, o Estágio S0 controla a ativação dos Blocos C0 e C1 a partir da condição da Entrada X0 – se estiver atuada, o Bloco C0 (composto pelos Estágios S10 e S11) será ativado; caso contrário, o Bloco C1 (composto apenas pelo Estágio S20) será ativado.

Ao ser ativado o Bloco C0 (X0 atuada), o Estágio S10 é automaticamente ativado, permanecendo nesta condição até que a Entrada X7 seja atuada (ativando o Estágio S11 e desativando o Estágio S10) ou a Entrada X0 não esteja mais atuada (desativando o Bloco C0 e ativando o Bloco C1). Da mesma forma, ao ser ativado o Bloco C1 (X0 não atuada), o Estágio S20 é automaticamente ativado, permanecendo nesta condição até que a Entrada X0 seja atuada (desativando o Bloco C1 e ativando o Bloco C0).

Neste capítulo foram apresentados os conceitos para programação de PLCs através da Linguagem Ladder, além dos tipos de dados disponíveis em uma CPU, as instruções básicas e a programação por estágios.

No próximo capítulo serão apresentados os conceitos básicos de sistemas supervisórios, incluindo um software comercial.

Capítulo 5

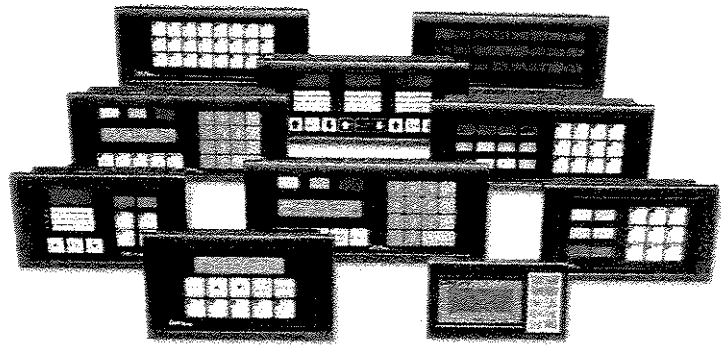
Sistemas Supervisórios em Automação Industrial – *Software Interact*

Este capítulo apresenta conceitos e campos de aplicação para os Sistemas Supervisórios, através da descrição de um Software Comercial (*Interact*). São utilizadas as seguintes referências bibliográficas: CTC (1997) e Stenerson (1999).

Inicialmente, os painéis de operação dos Sistemas Automatizados (Máquinas e Processos Industriais), implementados com PLCs, utilizavam 'Elementos Mecânicos' como Dispositivos de Entrada (Botoeiras, Chaves e Potenciômetros, por exemplo) e Saída (Lâmpadas, 'Buzinas'/ Sirenes, Indicadores Analógicos – *VU Meter*, por exemplo). Através destes era possível atuar no controle dos equipamentos e monitorar certas condições dos mesmos.

Com o desenvolvimento tecnlógico, tais elementos foram substituídos por Interfaces Homem-Máquina – IHMs (*HMI – Human Machine Interface*, *MMI – Man Machine Interface* ou ainda *OIT – Operator Interface Terminal*), que comunicam-se serialmente com o PLC, eliminando os custos dos Módulos de I/O utilizados para este fim no sistema anterior.

Atualmente, estas Interfaces podem ser compostas por uma área de visualização (Display, LEDs, etc) e/ou uma área de atuação (Teclado Numérico ou Alfa-numérico, Teclas de Funções, Teclas de Atuação, etc), conforme apresentado na figura 5.1.



Interfaces OptiMate – *PLC Direct by Koyo*

Figura 5.1 – Exemplo de Interfaces Homem-Máquina

A crescente competição industrial – caracterizada pela necessidade de aumento da produção, certificação da qualidade dos produtos e redução dos custos operacionais – exige mais eficiência dos Sistemas Automatizados de Produção, os quais devem proporcionar maior flexibilidade de operação e controle. Entre as características necessárias para atender tais requisitos incluem-se:

- Interface Amigável com Operador – facilidade de visualização e operação da Máquina ou Processo controlados
- Monitoramento e Gerenciamento de Condições de Alarme
- Utilização e Armazenamento de Programas/Receitas – alteração rápida de valores utilizados no controle, conforme o produto
- Armazenamento de Valores para Certificação dos Produtos – Histórico de Tendências
- Geração Automática de Relatórios – Controle Estatístico de Processo
- Acesso Automático a Bancos de Dados
- Acesso Compartilhado e Remoto – Conexão em Rede e através de Modem ou Rádio

Várias Interfaces Homem-Máquina, disponíveis atualmente no mercado, apresentam algumas destas características – Monitoramento de Alarmes e Armazenamento de Programas, por exemplo. No entanto, não atendem completamente às necessidades de "Gerenciamento do Sistema" – Histórico de Tendências, Relatórios, Acesso a Bancos de Dados, Conexão em Rede e/ou Remota, por exemplo. Nestes casos, ou simplesmente por se desejar uma Interface Homem-Máquina mais sofisticada e poderosa, utilizam-se os Sistemas Supervisórios.

5.1 Sistemas Supervisórios

Também conhecidos como *SCADA* (*Supervisory Control And Data Acquisition* – Controle Supervisório e Aquisição de Dados), os Sistemas Supervisórios constituem uma importante ferramenta da Automação Industrial, especialmente em aplicações que necessitem de Sistemas de Informação (Manipulação e Gerenciamento de Dados).

Tratam-se de Softwares dedicados, geralmente com 'Programação Orientada ao Objeto', que além das caracteríticas de IHM, proporcionam recursos avançados para Gerenciamento da Informação. Tais características são encontradas no Software apresentado a seguir, que reúne os principais recursos dos Sistemas Supervisórios disponíveis no mercado.

5.2 Software *Interact*

Trata-se de um Software Supervisório para plataforma PC, voltado para o segmento de Máquinas (*OEM – Original Equipment Manufacturer*) desenvolvido pela empresa americana CTC – Computer Technology Corporation (Milford, Ohio), a qual foi incorporada à multinacional Parker Hannifin no início de 1998, tornando-se assim uma divisão desta – CTC Parker Automation.

Tendo como principal foco o mercado de Máquinas, nem sempre suas características atendem a todas as necessidades de aplicações na área de Processos. Porém, apresenta a maioria dos recursos encontrados nos Softwares Supervisórios disponíveis para tal área.

5.2.1 Ambiente de Trabalho

O Softwatre *Interact* (V5.21 – fevereiro/99) é um aplicativo de 32 bits, desenvolvido para rodar em Windows® 95 ou NT.

No entanto, possui dois ambientes de Operação:

- Desenvolvimento em ambiente Windows® – a interface gráfica possibilita a criação de Painéis de Operação/Monitoramento e a configuração dos Módulos de Gerenciamento de forma fácil e rápida

- *Runtime* (Execução) em ambiente DOS® – além de maior segurança, este ambiente proporciona maior velocidade de comunicação com PLC

Assim, a Aplicação é desenvolvida em ambiente Windows®, conforme apresentado na figura 5.2, e executada em ambiente DOS®, conforme apresentado na figura 5.3.

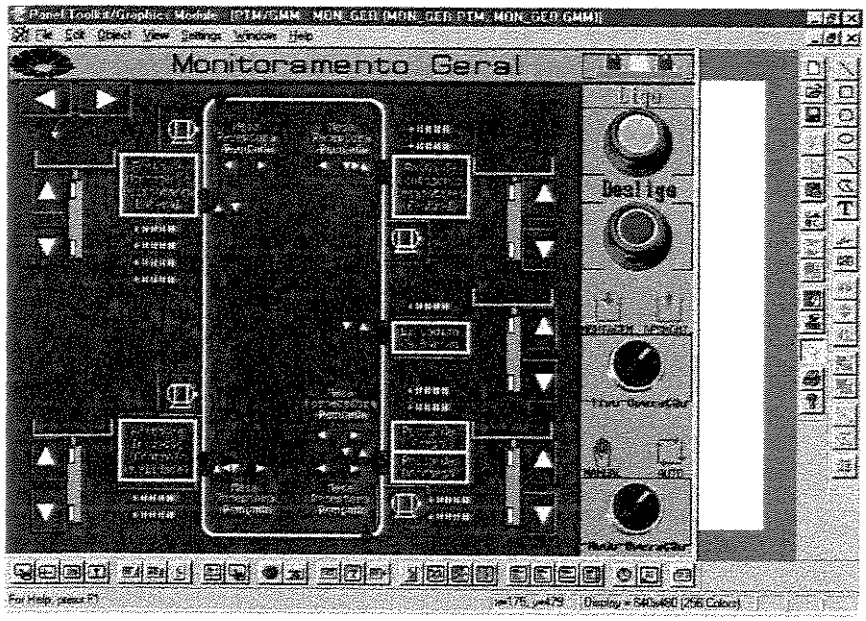


Figura 5.2 – *Interact* Desenvolvimento (Ambiente Windows®)

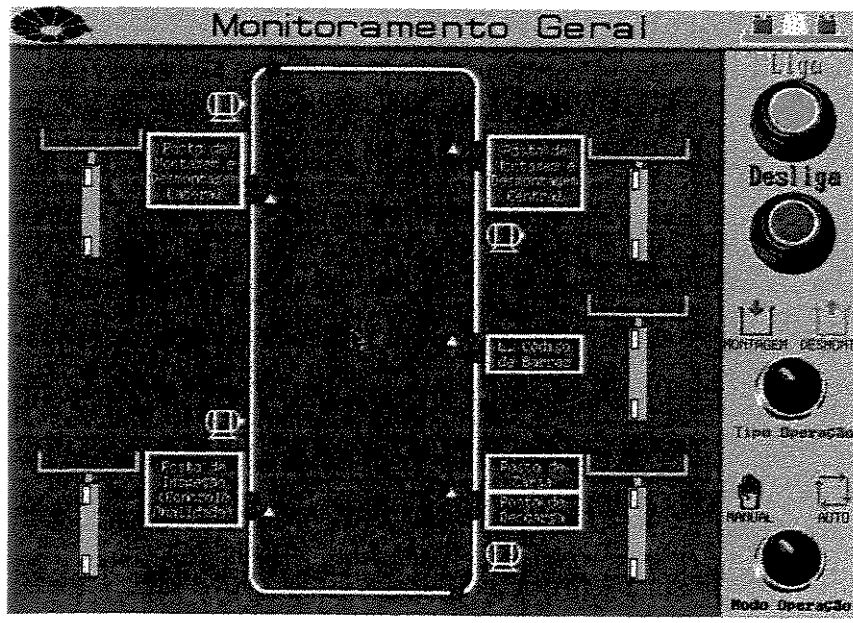


Figura 5.3 – *Interact Runtime* (Ambiente DOS®)

5.2.2 Chaves de Segurança – *Hard Keys*

Como a maioria dos Softwares Supervisórios, o *Interact* também possui uma Chave de Segurança - *Hard Key* (Trava de Hardware).

A *Hard Key* determina quais os Módulos do *Interact* podem ser utilizados, tanto para o Desenvolvimento quanto para o *Runtime*. A habilitação futura de Módulos é feita através de um código fornecido pela CTC Parker Automation, após a aquisição do mesmo.

Se a *Hard Key* estiver habilitada apenas com Módulos *Runtime*, não é possível utilizá-la para o desenvolvimento da Aplicação. No entanto, se estiver habilitada com Módulos de Desenvolvimento, pode ser utilizada também para executar a Aplicação.

A figura 5.4 apresenta a *Hard Key* para o Software *Interact*, a qual deve ser conectada à saída paralela do PC, devendo permanecer conectada durante todo o tempo de utilização do Software.

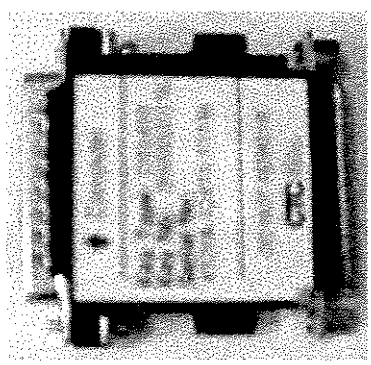


Figura 5.4 – *Hard Key (Interact)*

5.2.3 Arquitetura do Software *Interact*

O Software *Interact* tem uma arquitetura modular, possuindo três componentes básicos, conforme apresentado na figura 5.5:

- O Gerenciador Central
- Os Módulos
- Os Drivers de Comunicação

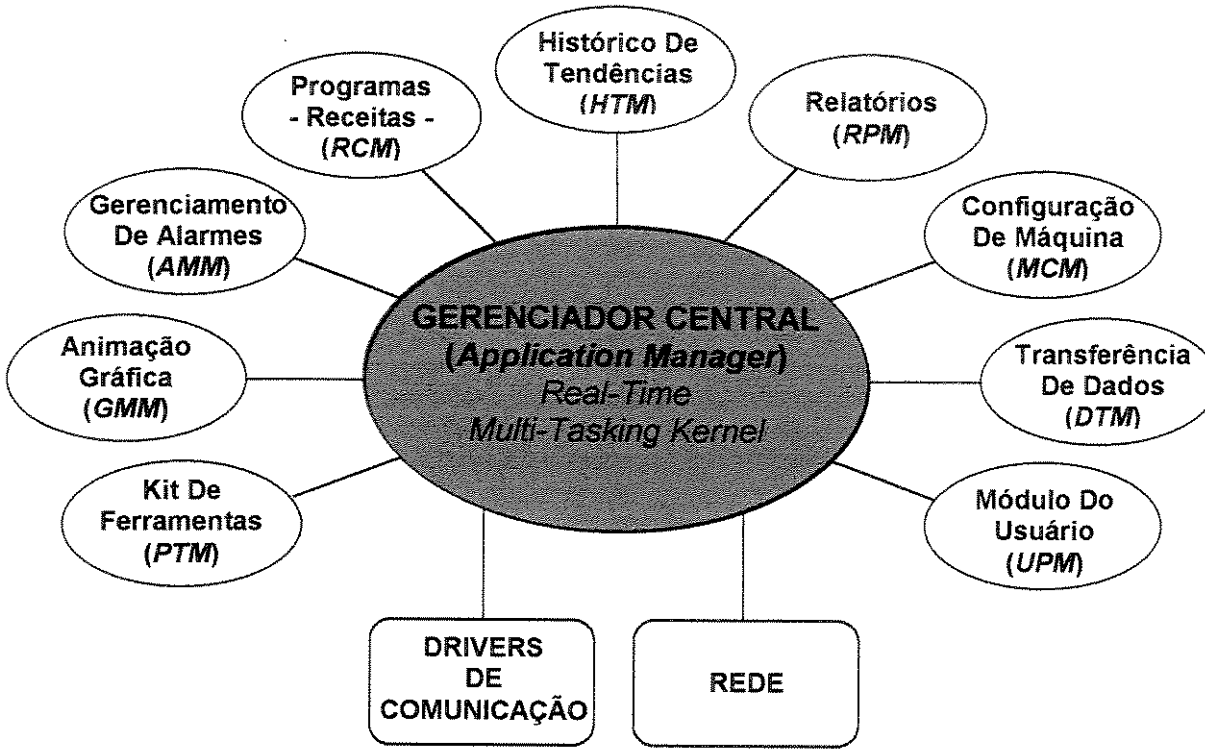


Figura 5.5 – Arquitetura *Interact*

Gerenciador Central (APM – Application Manager)

É o "coração" do *Interact*. Trata-se de um *kernel* ('núcleo') multi-tarefas em tempo real. Durante a execução do Software, é responsável pelo gerenciamento da Aplicação: controla o fluxo de dados entre os Módulos e os Drivers de Comunicação, incluindo todas as atividades, desde a apresentação de painéis ao controle do armazenamemnto de dados (*data logging*).

A eficiência de um Sistema Supervisório depende diretamente do Gerenciador Central. A figura 5.6 ilustra a relação entre o Gerenciador Central e os demais componentes do *Interact*.

Módulos

São componentes opcionais que proporcionam características específicas ao Sistema Supervisório (recursos/funcionalidades). Os Módulos utilizados dependem das necessidades específicas de cada Aplicação.

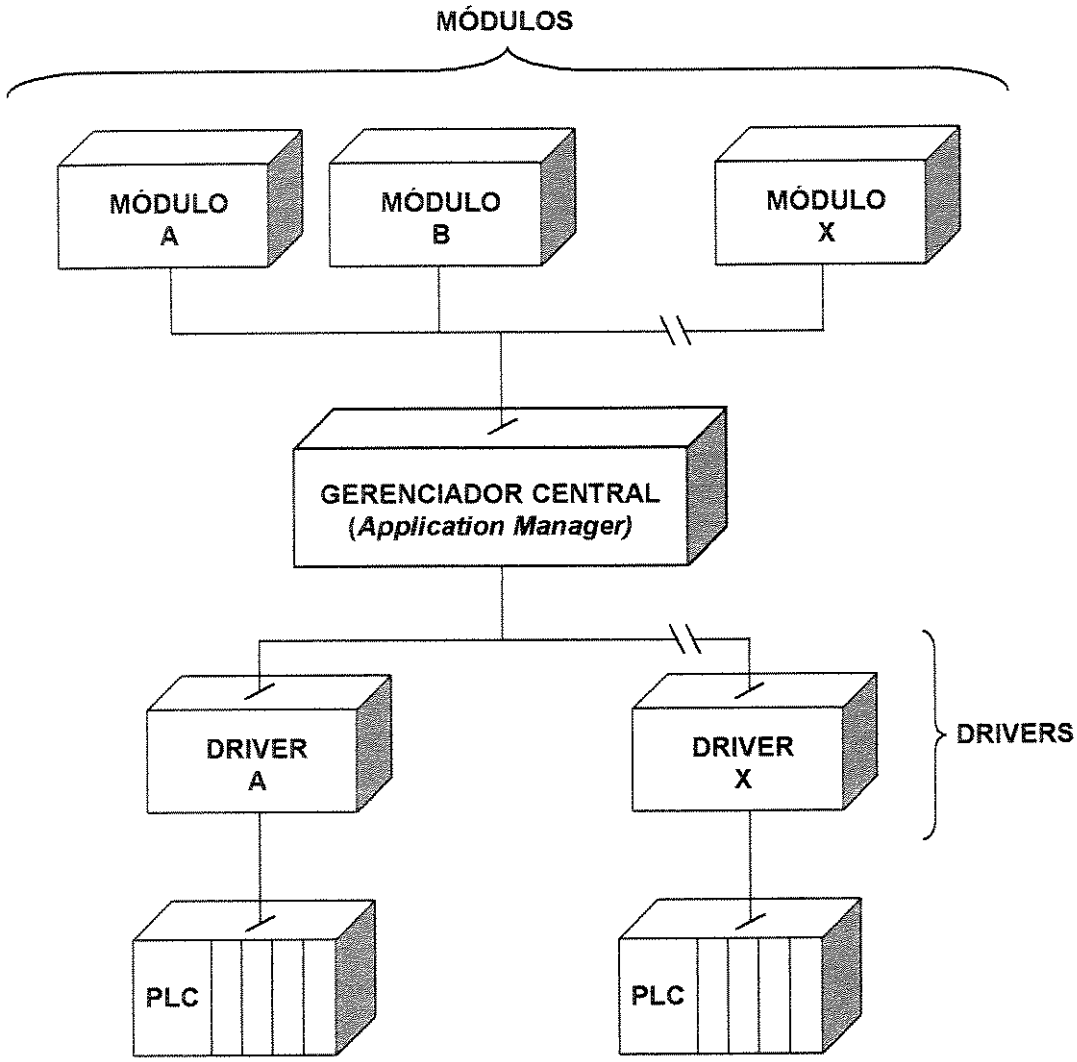


Figura 5.6 – Gerenciador Central (*Application Manager*)

Atualmente, os seguintes Módulos estão disponíveis no Software *Interact*:

- **Kit de Ferramentas do Painel (*PTM – Panel ToolKit Module*)**: Permite a construção fácil e rápida de Painéis de Operação e Gerenciamento compostas por Ferramentas de Entrada e Saída, tais como: *Push Buttons*, *Meters*, Displays Numéricos, Displays de Mensagens, *Bar Graphs*, *Ploters XY*, Teclas de Funções, entre outros. As Ferramentas utilizadas são configuradas para enviar e/ou receber dados (discretos ou analógicos) do PLC. A figura 5.7 apresenta um Painel de Operação construído apenas com o Módulo PTM.

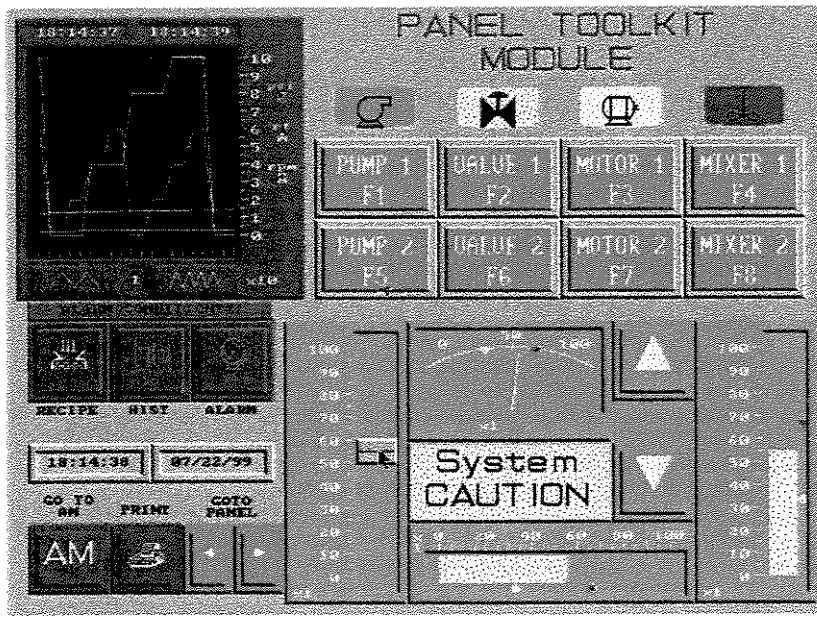


Figura 5.7 – Módulo *Interact – Kit de Ferramentas do Painel (PTM)*

- **Animação Gráfica (*GMM – Graphic Monitoring Module*)**: É utilizado em conjunto com o PTM, permitindo a representação e animação gráfica do Sistema Automatizado. Possui um editor gráfico para criação de objetos e imagens. Também, permite importar figuras em até 256 cores nos principais formatos (JPG, BMP, TIF, PCX, WMF, ICO, TGA e EPS) e figuras CAD (DXF), além de proporcionar conexão direta a Scanners. A figura 5.8 apresenta um Painel com os Módulos PTM e GMM.

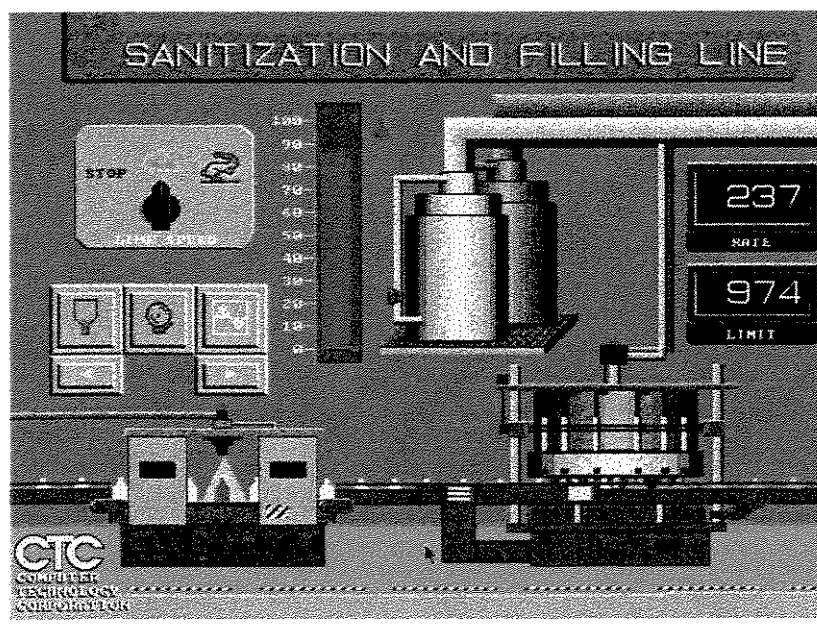


Figura 5.8 – Módulos *Interact – Kit de Ferramentas do Painel + Animação Gráfica (PTM / GMM)*

- **Gerenciamento de Alarmes (*AMM – Alarm Management Module*)**: Os alarmes configurados durante o Desenvolvimento são monitorados continuamente durante o *Runtime*. As Condições de Alarme detectadas são apresentadas em uma Tela específica (*Alarm Summary Display* – Apresentação do Resumo de Alarmes), através de uma linha de texto que pode conter os seguintes dados:
  - data e horário que o alarme ocorreu
  - data e horário que o alarme se tornou inativo
  - data e horário que o alarme foi reconhecido
  - descrição da condição de alarme
  - valor do alarme (no momento em que o mesmo ocorreu)
  - valor atual da condição monitorada
  - nível de prioridade do alarme
  - grupo a que pertence o alarme
  - nome do operador que reconheceu o alarme

Esta Tela permite a apresentação de uma mensagem ao operador (procedimento) relativa ao alarme ativo. Todas as informações gerenciadas pelo AMM podem ser aramazenadas automaticamente em arquivo (*data logging* – registro de dados) para posterior análise. A figura 5.9 apresenta o *Alarm Summary Display* do AMM.

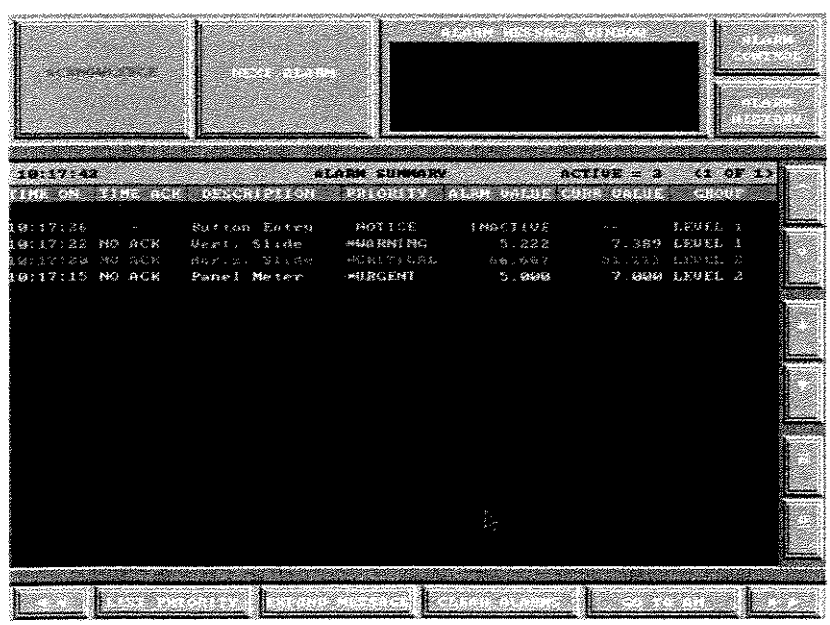


Figura 5.9 – Módulo *Interact – Gerenciamento de Alarmes (AMM)*

- **Programas / Receitas (*RCM – Recipe Module*)**: Durante o Desenvolvimento são criadas as receitas básicas que serão utilizadas em *Runtime*. Uma receita é um conjunto de variáveis, com valores específicos relativos a determinado produto, a qual é utilizada para a produção do mesmo. Durante o *Runtime*, a receita é enviada (*download*) ao PLC para execução de determinado produto, podendo ser editada, originando uma nova receita (com título/produto próprio) que será incluída na lista disponível para utilização. Todas as tarefas possíveis de serem realizadas em *Runtime* podem ser protegidas por nível de senha, permitindo apenas que operadores habilitados possam alterar, criar e enviar receitas, por exemplo. As atividades (criação, edição, transferência, etc) relativas a cada receita podem ser armazenas automaticamente em arquivo (*data logging* – registro de dados) para posterior análise. A figura 5.10 apresenta a Tela principal do RCM.

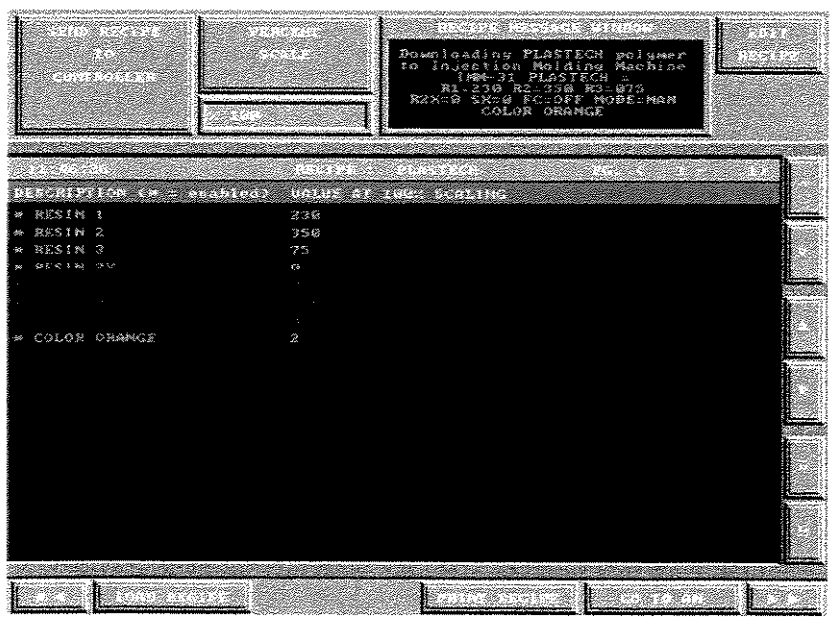


Figura 5.10 – Módulo *Interact – Programas / Receitas (RCM)*

- **Histórico de Tendências (*HTM – Historical Trending Module*)**: Em Desenvolvimento são configuradas as variáveis que serão monitoradas constantemente em *Runtime*, sendo aramazenadas em arquivo (*data logging* – registro de dados). O HTM proporciona a visualização das variáveis monitoradas em formato gráfico (curvas) e texto (relatório), além de permitir a conversão do arquivo gerado para formato ASCII, para análise em Planilhas Eletrônicas, por exemplo. Conta também com recursos de busca e análise das informações

registradas. Cada arquivo pode monitorar (registrar) até 16 variáveis, com a possibilidade de criação de arquivos distintos para monitoramento de maior número de variáveis. A figura 5.11 apresenta a Tela do HTM para visualização em formato gráfico.

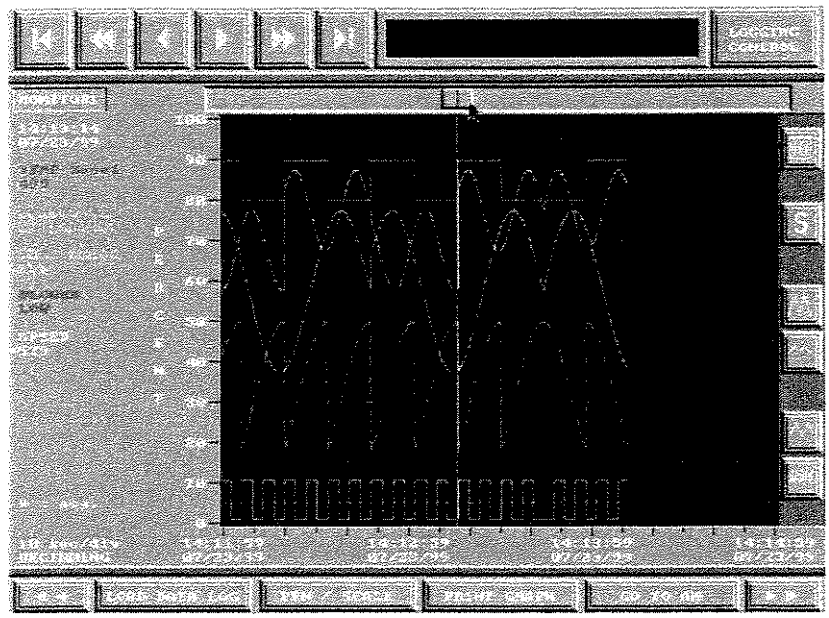


Figura 5.11 – Módulo *Interact – Histórico de Tendências (HTM)*

- **Relatórios (*RPM – Report Module*)**: O formato do Relatório e as variáveis a serem monitoradas, em tempo real em *Runtime*, são configuradas durante o Desenvolvimento. Além do monitoramento e registro dos dados, possui recursos estatísticos que podem ser utilizados para Controle Estatístico de Processo. Permite que o relatório gerado seja gravado em arquivo e/ou impresso. A figura 5.12 apresenta o formato de um relatório criado com o RPM.

- **Configuração de Máquina (*MCM – Machine Configuration Module*)**: Permite a transferência de parâmetros (através de listas de variáveis com valores pré-definidos) que determinam a configuração da máquina. Com isso, facilita tarefas geralmente utilizadas pelos Fabricantes de Máquina (*OEMs*), que habilitam determinadas funções ou recursos conforme o modelo da máquina, sem alterar o Programa de Aplicação (PLC). Toda e qualquer transferência de parâmetros é permitida, monitorada e registrada (*data logging*) através de níveis de senha. A figura 5.13 apresenta a Tela principal do MCM.

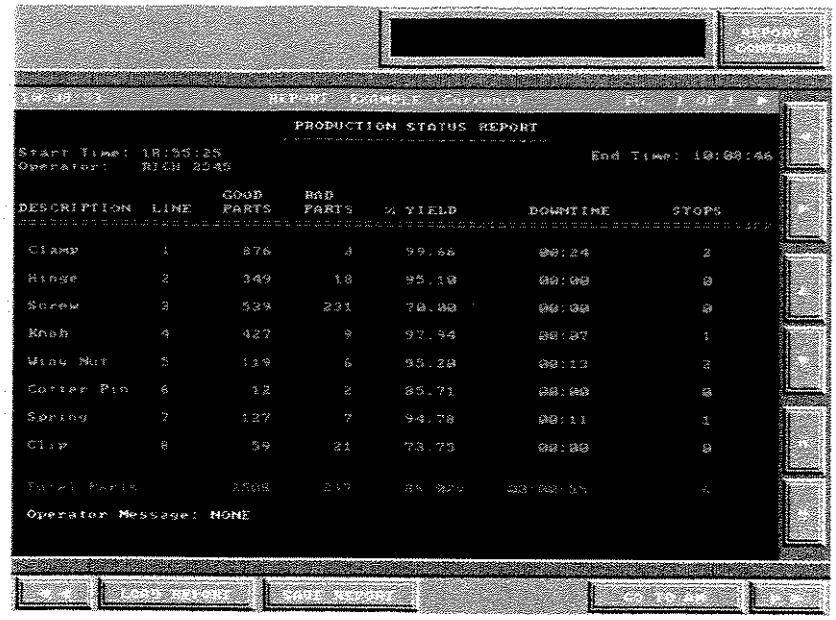


Figura 5.12 – Módulo *Interact – Relatórios (RPM)*

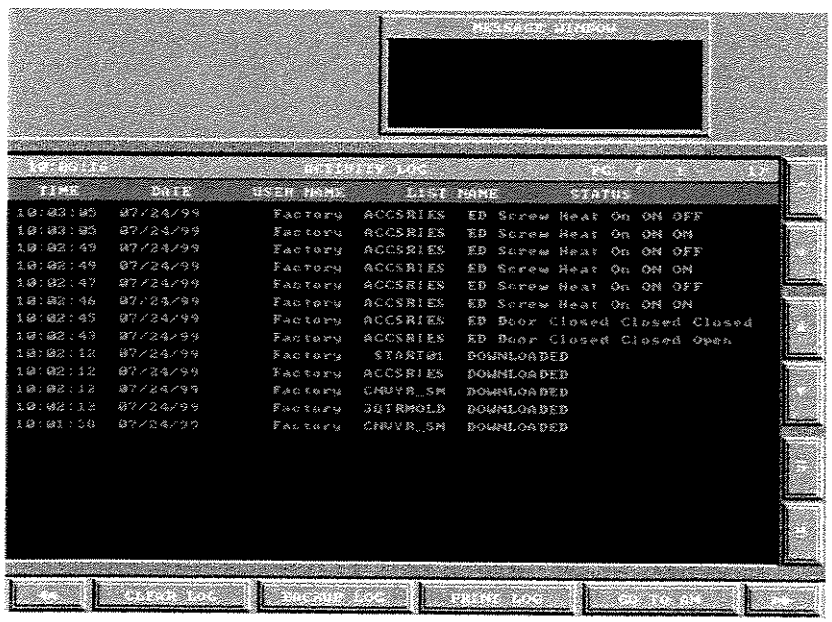


Figura 5.13 – Módulo *Interact – Configuração de Máquina (MCM)*

- **Transferência de Dados (*DTM – Data Transfer Module)***: Permite a transferência de dados entre dispositivos distintos (PLCs de fornecedores distintos ou múltiplos PLCs de um mesmo fornecedor) sem necessidade de hardware extra. Em Desenvolvimento, são configurados os endereços que serão transferidos de um PLC a outro, bem como o controle de cada transferência. Em *Runtime*, O DTM não possui Tela específica, uma vez que

apenas gerencia a transferência dos dados, conforme configurado. A figura 5.14 apresenta o esquema funcional do DTM.

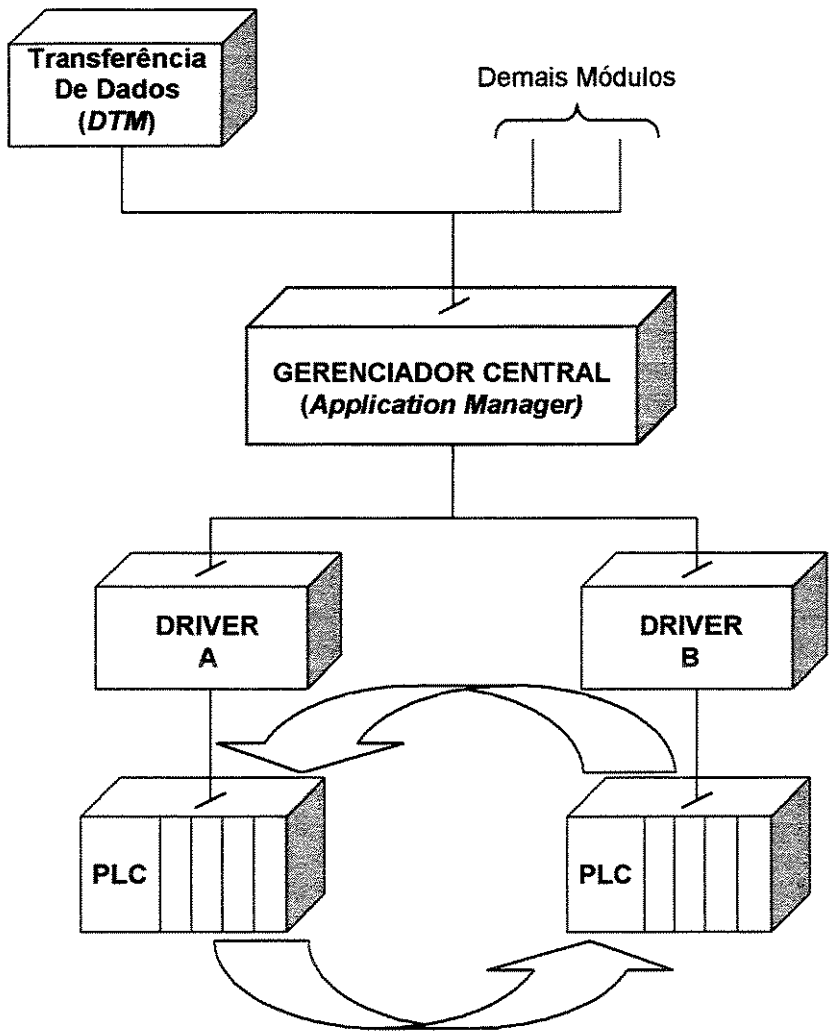


Figura 5.14 – Módulo *Interact – Transferência de Dados (DTM)*

- **Módulo do Usuário (*UPM – User Program Module)***: Se a Aplicação necessitar de alguma característica não disponível nos demais Módulos, utiliza-se o UPM, através do qual o usuário configura seu próprio Módulo para o *Interact*. Através de um Programa Residente (*TSR – Terminate Stay Resident*) – normalmente desenvolvido em Linguagem 'C', o usuário confere ao *Interact* as características e recursos específicos que necessita. O UPM comunica-se com o Programa do Usuário através de interrupções de software, e este

comunica-se com outros Módulos e Drivers do *Interact* através de uma área de memória compartilhada (*shared memory*), confome apresentado na figura 5.15.

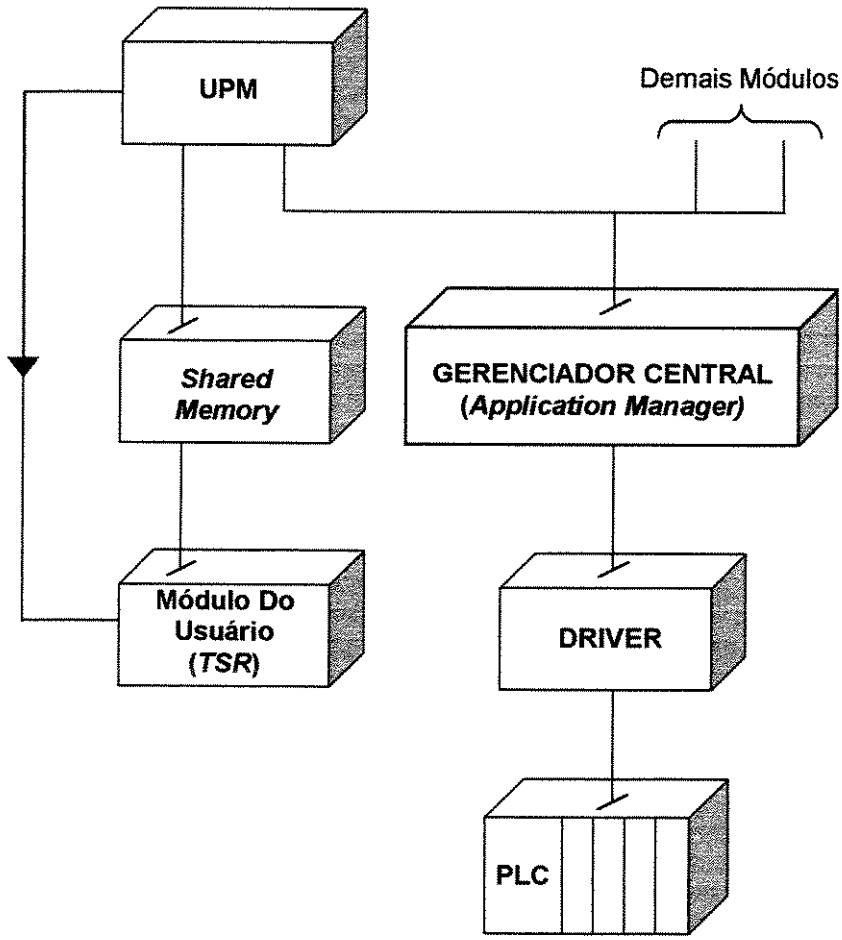


Figura 5.15 – Módulo *Interact – Módulo do Usuário (UPM)*

Os Módulos PTM, GMM e AMM fazem parte do 'Pacote Básico do *Interact*'. Os demais Módulos são opcionais, sendo fornecidos separadamente conforme a necessidade de cada Aplicação.

Todos os Módulos opcionais, inclusive o AMM, podem ser acessados/operados através de Painéis do PTM por meio de *Links* (variáveis reservadas com funções pré-definidas pelo *Interact*).

Por exemplo, pode-se verificar a relação de alarmes ocorridos sem acessar a Tela específica do AMM, através de Ferramentas do PTM (Painel de Mensagens) configuradas com *links* do Módulo de Gerenciamento de Alarmes.

Drivers de Comunicação

São programas (Softwares) específicos usados para transferir dados entre dispositivos ou equipamentos. Têm a função de padronizar o formato dos dados entre dispositivos não-similares, tornando-os compatíveis. Controlados pelo Gerenciador Central, permitem que uma Aplicação seja configurada para acessar qualquer dispositivo (PLC, por exemplo) por meio de protocolo apropriado, proporcionando aos Módulos utilizados o acesso em tempo real sobre condições/operações do Sistema Automatizado monitorado. A figura 5.16 apresenta a comunicação entre o Gerenciador Central e o PLC, através de um Driver apropriado.

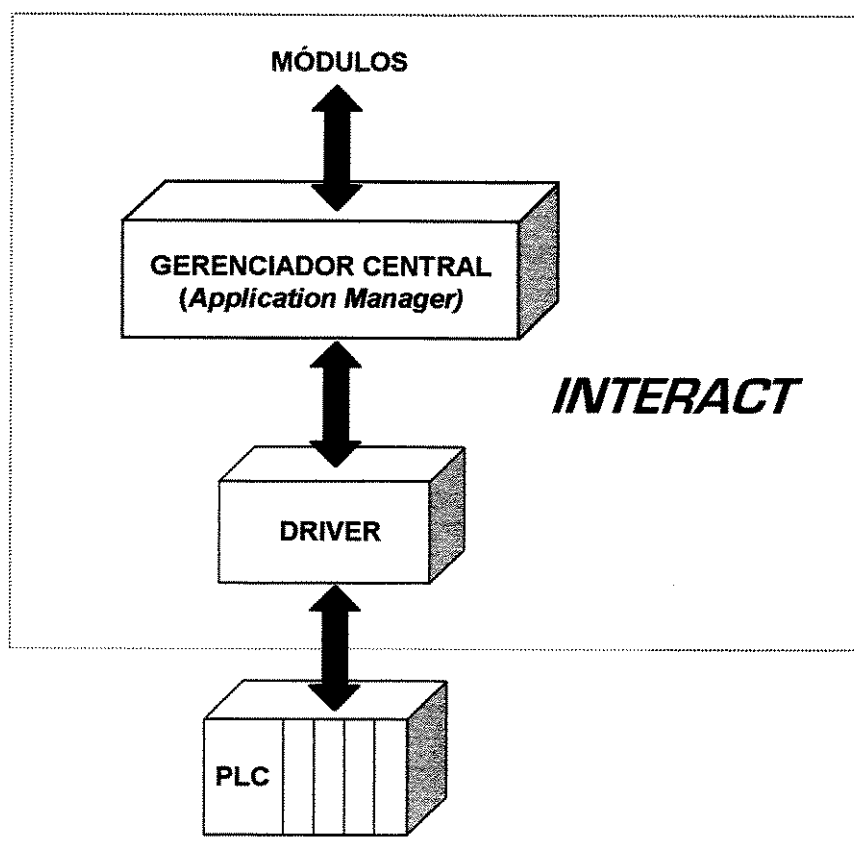


Figura 5.16 – Driver de Comunicação

Há dois tipos básicos de Drivers de Comunicação:

- drivers seriais, que acessam o PLC diretamente através da porta serial do PC (RS-232)
- drivers que necessitam de interfaces inseridas no PC para comunicação com o PLC

Atualmente, o *Interact* possui Driver de Comunicação para mais de 45 dispositivos. Além de Drivers para os PLCs mais utilizados nos Estados Unidos e na Europa, possui o Driver do Usuário (*UDI – User Device Interface*), que permite o desenvolvimento de novos Drivers de Comunicação para necessidades específicas (normalmente em Linguagem 'C').

Todos os Drivers de Comunicação com dispositivos externos fazem parte do 'Pacote Básico do *Interact*', sendo que dois ou mais Drivers podem ser utilizados simultaneamente (Sistema Supervisório conectado a diferentes tipos de equipamentos). No entanto, possui dois Drivers opcionais que são fornecidos separadamente, conforme a necessidade da Aplicação:

- *NBIOS – NetBIOS Network Driver*, que permite a conexão do *Interact* em rede (acesso remoto e/ou compartilhado)
- *IMD – Interact Modem Driver*, que permite acesso ao *Interact* através de Modem

Com esses Drivers, o *Interact* pode ser acessado remotamente por outro(s) PC(s) executando:

- Aplicações *Interact*
- Outro Sistema Supervisório qualquer que possibilite comunicação em rede
- Aplicativos Windows® (Excel ou Visual Basic, por exemplo), através do *Interact DDE Server* – fornecido separadamente pela CTC Parker Automation

5.2.4 Princípio de Funcionamento

O *Interact* interage com os dispositivos externos através de uma relação *Master-Slave* (Mestre-Escravo), assumindo a função de *Master*. Todas as operações de escrita e leitura devem ser requisitadas pelo *Interact*, antes que a transferência de dados seja iniciada, conforme descrito a seguir.

Escrita de Dados

A Escrita de Dados (dados enviados ao PLC), conforme apresentado na figura 5.17, respeita a seguinte seqüência de operação:

- uma mensagem é formatada com todas as informações necessárias, e enviada ao PLC de destino
- o PLC retorna um sinal de reconhecimento (*ACK – Acknowledge*) para o *Interact*, indicando que a mensagem de escrita foi recebida
- com o dado escrito, o PLC envia uma mensagem ao *Interact* indicando que a operação foi finalizada
- *Interact* recebe a mensagem e retorna um sinal *ACK* ao PLC, indicando tal recebimento

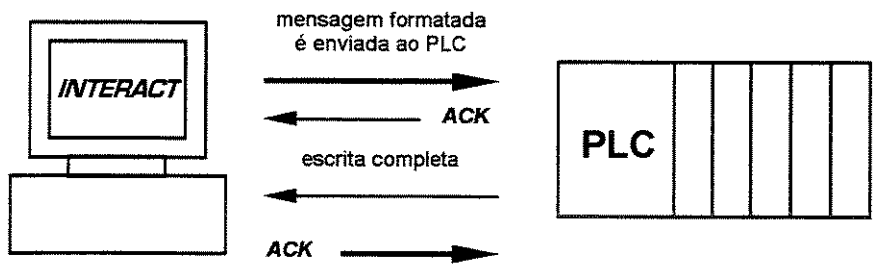


Figura 5.17 – Operação de Escrita de Dados

Leitura de Dados

A Leiturade Dados (dados enviados pelo PLC), conforme apresentado na figura 5.18, respeita a seguinte seqüência de operação:

- uma mensagem é formatada pelo *Interact* e enviada ao PLC de destino requisitando os dados
- o PLC retorna um sinal *ACK* para o *Interact*, indicando que a mesnsagem de leitura foi recebida
- o dado solicitado é colocado em uma mensagem, a qual é enviada ao *Interact*
- *Interact* recebe a e retorna um sinal *ACK* ao PLC, indicando tal recebimento

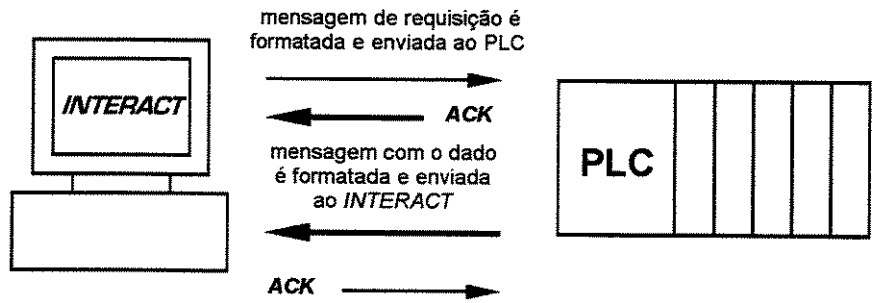


Figura 5.18 – Operação de Leitura de Dados

Ao contrário da maioria dos Softwares Supervisórios, o *Interact* não apresenta limite fixo para quantidade de variáveis de escrita/leitura. Os fatores limitantes nesta questão são:

- quantidade de memória para armazenamento da Aplicação (HD)
- processador utilizado pelo sistema (mínimo 80286)

5.3 Configuração dos Itens

Independente da operação a ser realizada, o formato de configuração dos itens de cada Módulo do *Interact* é idêntica. Utiliza-se um formato padrão para Operações de Escrita ou de Leitura, como apresentado na figura 5.19. Os caracteres apresentados como delimitadores podem variar conforme o Driver utilizado.

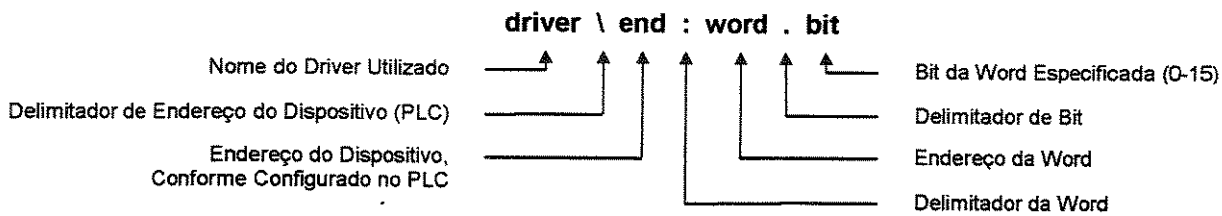


Figura 5.19 – Configuração Endereços de Escrita/Leitura

Portanto, o que determina se a Operação será de Escrita ou de Leitura não é o formato utilizado, e sim o item configurado (Ferramenta de Escrita ou de Leitura, por exemplo).

É possível realizar operações trigonométricas, aritméticas, lógicas e de comparações com as variáveis lidas/escritas, através de funções e operadores:

SIN(x) seno de x

COS(x) cosseno de x

TAN(x) tangente de x

ASIN(x) arco-seno de x

ACOS(x) arco-cosseno de x

ATAN(x) arco-tangente de x

LOG(x) logarítmo de x

NLOG(x) logarítmo natural de x

EXP(x) exponenciação de 'e' ('e' elevado a x)

SQRT(x) raiz quadrada de x

ABS(x) valor absoluto de x

NOT(x) complemento '1' de x

( ) parênteses: (x + y) / z

+ adição: x + y

- subtração: x – y

* multiplicação: x * y

/ divisão: x / y

% módulo da divisão: x % y (por exemplo, 7 % 2 = 3, e resto 1 => módulo 1)

^ exponenciação: x ^ y (por exemplo, 3 ^ 3 = 27)

& lógica AND (bit)

| lógica OR (bit)

~ lógica XOR (bit)

&& lógica AND (word)

|| lógica OR (word)

<< deslocamento (*shift*) à esquerda (x << y => um número binário x é deslocado à esquerda por y posições)

>> deslocamento (*shift*) à direita (x >> y => um número binário x é deslocado à direita por y posições)

= comparação de igualdade

< comparação de inferioridade

> comparação de superioridade

<= comparação de inferioridade ou igualdade

>= comparação de superioridade ou igualdade

<> comparação de diferença

Por exemplo, uma Ferramenta de Escrita configurada como **dvr \ 1 : 2000 = ? * 1000**, determina que a variável (word) **2000**, do dispositivo **1**, acessado pelo driver **dvr**, receberá o valor fornecido pelo operador (**?**) multiplicado por **1000**.

E uma Ferramenta de Leitura configurada como **sin (dvr \ 1 : 2010)**, determina que esta Ferramenta apresentará o seno (**sin**) do valor da variável (word) **2010**, do dispositivo **1**, acessado pelo driver **dvr**.

Neste capítulo foram apresentados os conceitos e campos de aplicação para os Sistemas Supervisórios, com descrição dos recursos do Software *Interact*.

No próximo capítulo, será apresentado o Projeto PIPEFA, com descrição dos Postos e Sistemas que o compõem, como exemplo de aplicação para os conceitos abordados neste trabalho.

Capítulo 6

Exemplo de Aplicação – *Projeto PIPEFA*

Neste capítulo será apresentada a estrutura completa da Plataforma Industrial para Pesquisa, Ensino e Formação em Automação (PIPEFA). Será dada ênfase à constituição e operação do Sistema de Tansferência, descrevendo a implementação através de PLC e Sistema Supervisório Local, com aplicação dos conceitos anteriormente abordados. São utilizadas as seguintes referências bibliográficas: Martins (1998), PLC Direct (1997) e Rosário (1996).

O projeto PIPEFA foi desenvolvido para ensino, pesquisa e formação tecnológica na área de automação industrial, proporcionando o desenvolvimento de aplicações em automação que possibilitem melhoria de desempenho, redução de custos, flexibilidade de operação e aumento da qualidade em um sistema de manufatura. É composta por componentes industriais, tais como: PLC, cilindros e eletro-válvulas pneumáticas, sensores magnéticos, sensores ópticos, motores elétricos, leitor de código de barras e sistema supervisório (PC), caracterizando uma situação real.

A plataforma realiza operações típicas de Sistemas Automatizados de Produção, como carregamento, descarregamento, transferência, operações de montagem e desmontagem, e inspeção (controle de qualidade), além de proporcionar o gerenciamento total do sistema.

O projeto PIPEFA estabeleceu um acordo de cooperação entre o Laboratório de Automação Integrada e Robótica (LAR) da UNICAMP, o Instituto de Automação da Fundação CTI e o LIISI (Laboratorie d'Ingénierie Integrée des Systèmes Industriels) da França. Esse acordo permitiu o

aprofundamento de conhecimentos, realizações no ponto de vista material e sua aplicação em Sistemas Automatizados de Produção, dentro dos objetivos de geração de uma massa crítica de pesquisadores no domínio da Engenharia de Automação Integrada.

6.1 Descrição da PIPEFA

A Plataforma Industrial para Pesquisa, Ensino e Formação em Automação (PIPEFA) permite a confecção (montagem e desmontagem) de um produto genérico, constituído de uma placa base sobre a qual podem ser colocados cubos – tipo LEGO®, em três posições diferentes e em até dois níveis de montagem, conforme apresentado na figura 6.1.

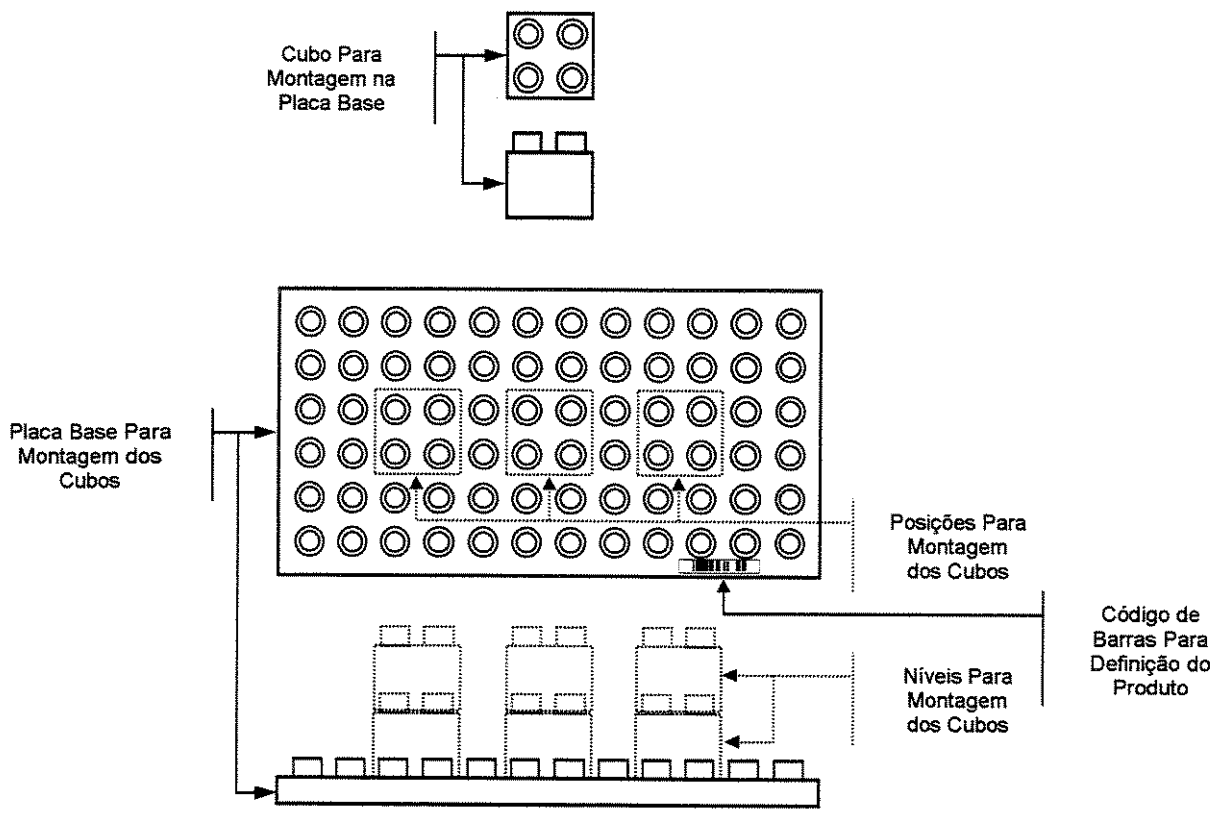


Figura 6.1 – Placa Base e Cubo LEGO® Utilizados na Confecção de Produtos pela PIPEFA

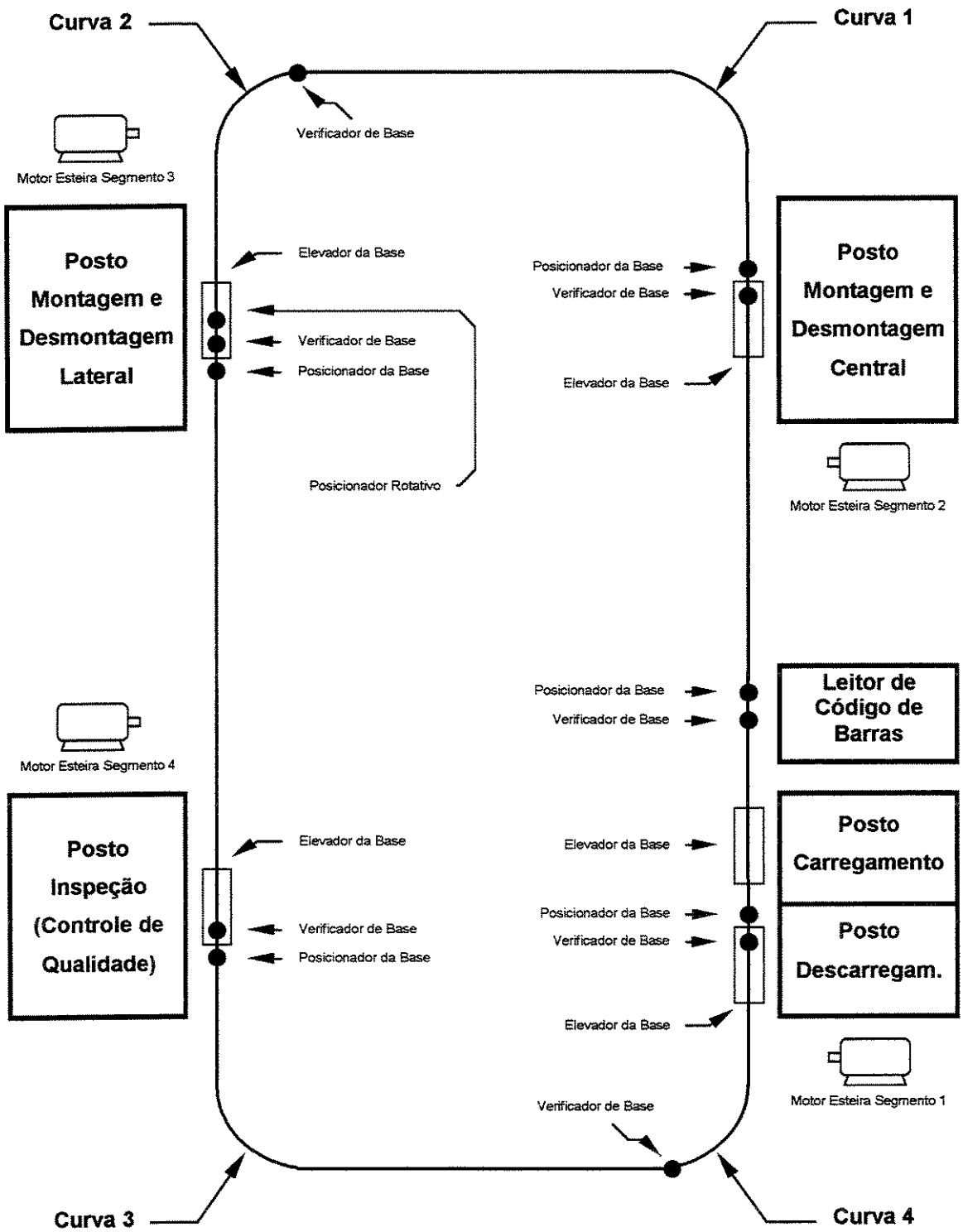


Figura 6.2 – Estrutura Básica PIPEFA

A figura 6.2 apresenta a estrutura básica da PIPEFA, a qual é composta por:

- Posto de Carregamento
- Leitor de Código de Barras
- Posto de Montagem/Desmontagem Central
- Posto de Montagem/Desmontagem Lateral
- Posto de Inspeção
- Posto de Descarregamento
- Sistema de Transferência, apresentado com detalhes na figura

A confecção de determinado produto inicia-se pelo fornecimento da placa base à esteira transportadora do Sistema de Transferência, realizado pelo Posto de Carregamento. De posse desta, a esteira irá transportá-la até o Leitor de Código de Barras, onde será reconhecido o produto a ser confeccionado, conforme o código presente na placa base, e definida a utilização dos Postos de Montagem Central e/ou Lateral. Finalizadas as operações de montagem, o produto confeccionado (placa base e respectivos cubos) é transportado ao Posto de Inspeção para verificação da montagem executada (Controle de Qualidade). Após esta operação, o produto é transportado ao Posto de Descarregamento, onde poderá ser rejeitado (produto não-conforme e irreparável), descarregado (produto 100% conforme) ou retornar aos Postos de Montagem Central e Lateral, sem interferência dos postos anteriores (produto não-conforme, mas possível de ser reparado), passando novamente pelo Posto de Inspeção.

Inicialmente cada Posto/Sistema da PIPEFA era controlado por um PLC específico. Atualmente, um único PLC pode controlar mais de um Posto. Por exemplo, um mesmo PLC controla o Sistema de Transferência, os Postos de Carregamento, de Descarregamento e de Inspeção, além do Leitor de Código de Barras. Um outro PLC controla os Postos de Montagem/Desmontagem Central e Lateral.

Os PLCs envolvidos no controle da PIPEFA estão interligados ao Sistema Supervisório Local (Software *Interact*), com finalidade de Interface Homem-Máquina baseada em PC, que proporciona controle e monitoramento das condições gerais de cada Posto. A figura 6.3 apresenta

a interligação entre os Postos Operacionais da PIPEFA e o Sistema Supervisório Local, com indicação do PLC usado em cada controle.

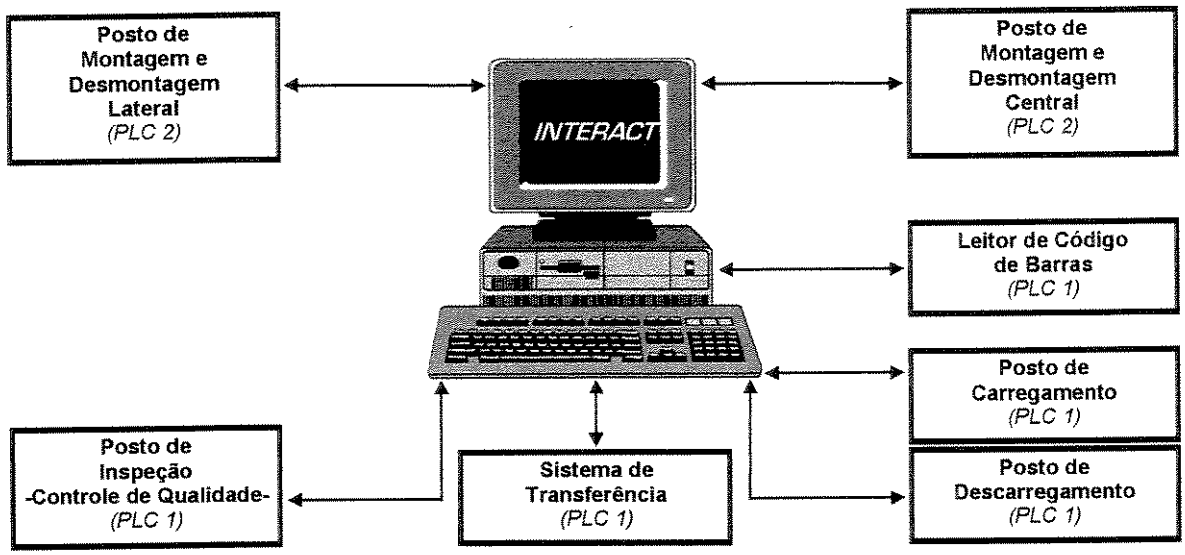


Figura 6.3 – Interligação dos Postos da PIPEFA ao Sistema Supervisório Local

A seguir, é feita a descrição de cada Posto/Sistema da PIPEFA, com ênfase ao Sistema de Transferência e ao Sistema Supervisório Local, focos principais deste trabalho.

6.1.1 Posto de Carregamento

O Posto de Carregamento, apresentado na figura 6.4, armazena as placas bases e as fornece à esteira transportadora para confecção do produto, através de sistema mecânico (com acionamento pneumático) que as posiciona sobre um elevador (Sistema de Transferência). Possui verificação da quantidade de placas bases armazenadas, que indica se o número mínimo foi atingido.

A operação deste Posto pode ser iniciada manualmente através do Sistema Supervisório Local, ou automaticamente pelo Sistema de Transferência.

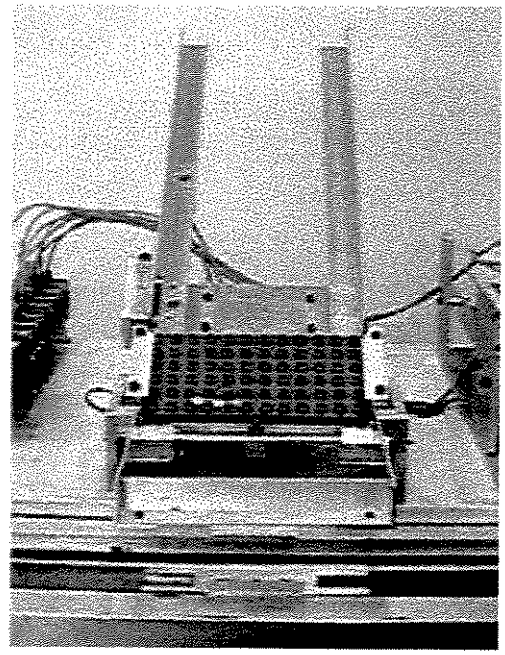


Figura 6.4 – Posto de Carregamento

6.1.2 Leitor de Código de Barras

Trata-se de um Leitor de Código de Barras comercial, conforme apresentado na figura 6.5, com saída serial (RS-232) conectado diretamente em uma das portas seriais do PLC. Faz a leitura do código de barras existente nas placas bases, definindo o produto a ser confeccionado. O código lido é enviado ao Sistema de Transferência, que determinará a utilização dos Postos de Montagem Central e/ou Lateral. O formato do código de barras presente nas placas bases encontra-se no Anexo II.

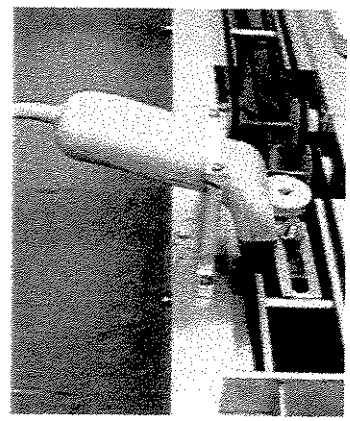


Figura 6.5 – Leitor de Código de Barras

A operação de leitura do código de barras somente é realizada se esta opção estiver habilitada no Sistema Supervisório Local. Caso contrário, o produto a ser confeccionado é determinado através de códigos fornecidos diretamente pelo operador.

6.1.3 Posto de Montagem/Desmontagem Central

Este Posto realiza as operações de montagem e desmontagem de cubos na posição central da placa base, nos dois níveis possíveis, conforme apresentado na figura 6.6.

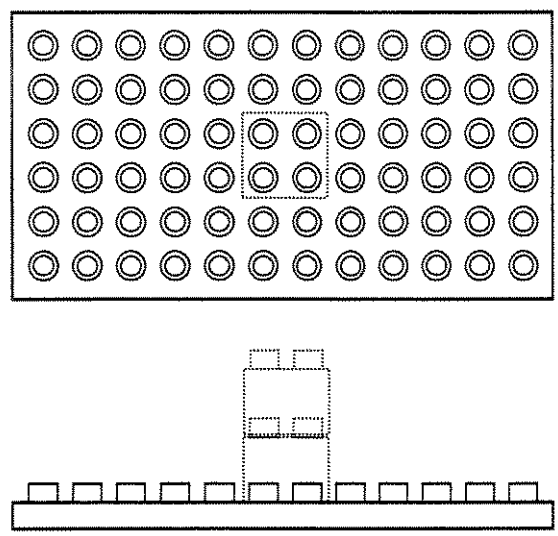


Figura 6.6 – Possibilidades de Operações no Posto de Montagem/Desmontagem Central

Através de sistema mecânico (com acionamento pneumático) recebe a placa base fornecida pelo elevador (Sistema de Transferência), a posiciona na área de atuação do sistema de montagem/desmontagem e executa as operações necessárias, retornando-a ao elevador que deposita-a novamente na esteira transportadora. Possui verificação da quantidade de cubos para montagem, que indica se o número mínimo foi atingido.

A operação deste Posto, apresentado na figura 6.7, pode ser realizada manualmente (passo a passo) através do Sistema Supervisório Local, ou automaticamente pelo Sistema de Transferência.

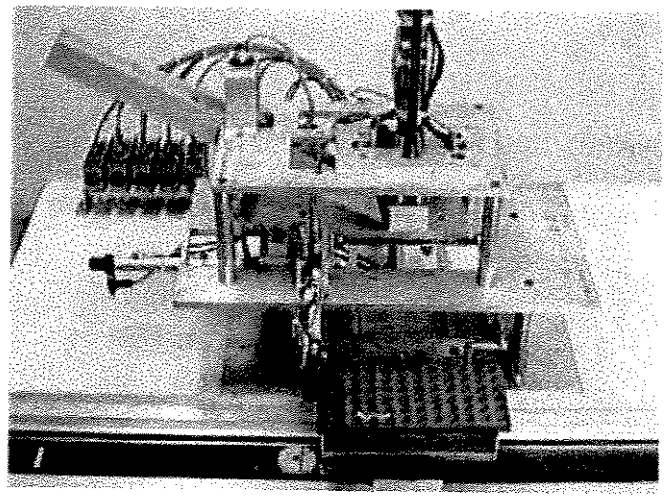


Figura 6.7 – Posto de Montagem/Desmontagem Central

6.1.4 Posto de Montagem/Desmontagem Lateral

Este Posto realiza as operações de montagem e desmontagem de cubos nas posições laterais (à direita e à esquerda) da placa base, nos dois níveis possíveis, conforme apresentado na figura 6.8.

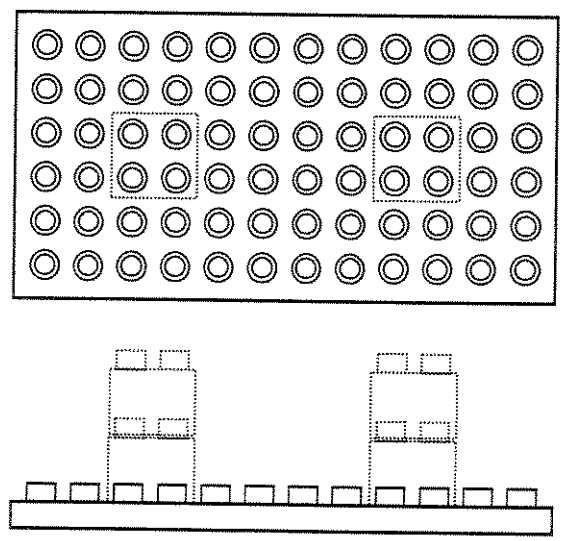


Figura 6.8 – Possibilidades de Operações no Posto de Montagem/Desmontagem Lateral

Através de sistema mecânico (com acionamento pneumático) recebe a placa base fornecida pelo elevador (Sistema de Transferência), a posiciona na área de atuação do sistema de montagem/desmontagem e executa as operações necessárias, retornando-a ao elevador que

deposita-a novamente na esteira transportadora. Possui verificação da quantidade de cubos para montagem, que indica se o número mínimo foi atingido.

A operação deste Posto, apresentado na figura 6.9, pode ser realizada manualmente (passo a passo) através do Sistema Supervisório Local, ou automaticamente pelo Sistema de Transferência.

Como pode ser observado na figura 6.9, o Posto de Montagem/Desmontagem Lateral possui sistema de montagem/desmontagem apenas à direita. Para montagem/desmontagem à esquerda, o elevador (Sistema de Transferência) possui um cilindro rotativo que rotaciona a placa base em 180°, iniciando nova seqüência de operações. Caso não seja necessário realizar montagem/desmontagem à direita, a placa base é rotacionada antes de ser recebida pelo Posto. Caso contrário, realiza-se montagem/desmontagem à direita e ao retornar a placa base ao elevador, este a rotaciona e a fornece novamente ao Posto para realização de montagem/desmontagem à esquerda. Nos dois casos, antes de ser depositada na esteira transportadora, a placa base é rotacionada novamente em sentido contrário, retornando à posição inicial.

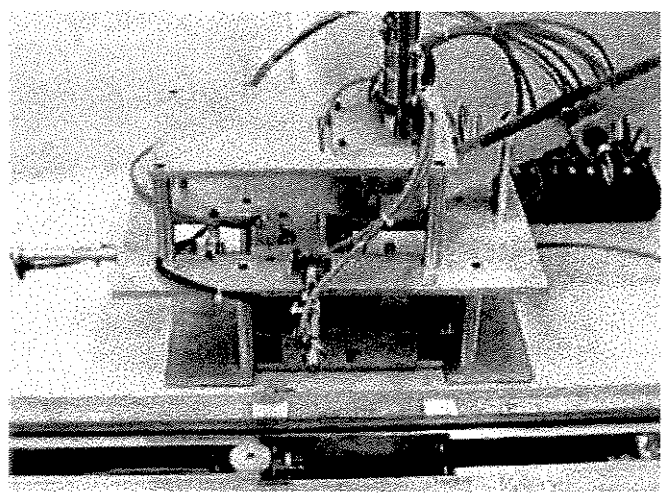


Figura 6.9 – Posto de Montagem/Desmontagem Lateral

Caso não seja necessário montagem/desmontagem à esquerda, a seqüência de operações se dá como no Posto de Montagem/Desmontagem Central.

6.1.5 Posto de Inspeção

Este Posto, apresentado na figura 6.10, realiza a inspeção do produto confeccionado, e envia o resultado ao Sistema de Transferência. Através de sistema mecânico (com acionamento pneumático) recebe o produto confeccionado (placa base e respectivos cubos) fornecido pelo elevador (Sistema de Transferência), posiciona-o na área de atuação do sistema de inspeção e, através de sensores ópticos, verifica a montagem (existência) de cubos em cada posição possível. Finalizada a inspeção, retorna o produto confeccionado ao elevador, que deposita-o novamente na esteira transportadora. A operação deste Posto pode ser iniciada manualmente através do Supervisório Local, ou automaticamente pelo Sistema de Transferência.

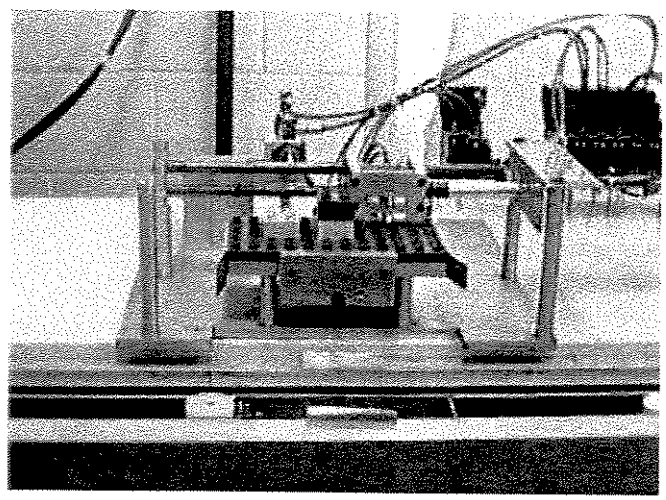


Figura 6.10 – Posto de Inspeção

Atualmente, a inspeção restringe-se ao posicionamento dos cubos apenas. Futuramente, com a instalação de sensores de cor neste Posto e nos Postos de Montagem/Desmontagem Central e Lateral será possível verificar também a cor dos cubos montados, aumentando consideravelmente a gama de produtos possíveis de serem confeccionados na PIPEFA.

6.1.6 Posto de Descarregamento

Através de sistema mecânico (com acionamento pneumático), o Posto de Descarregamento recebe o produto confeccionado fornecido pelo elevador (Sistema de Transferência), para ser rejeitado (produto não-conforme e irreparável) ou descarregado (produto 100% conforme), sendo

armazenado no próprio Posto. Caso o produto tenha sido classificado como não-conforme, mas com possibilidade de ser reparado, o Posto de Descarregamento não é acessado e o produto é transportado diretamente aos Postos de Montagem/Desmontagem Central e/ou Lateral, passando posteriormente por nova inspeção. Possui verificação da quantidade de produtos armazenados, que indica se o número máximo foi atingido. A operação deste Posto, apresentado na figura 6.11, pode ser iniciada manualmente através do Sistema Supervisório Local, ou automaticamente pelo Sistema de Transferência.

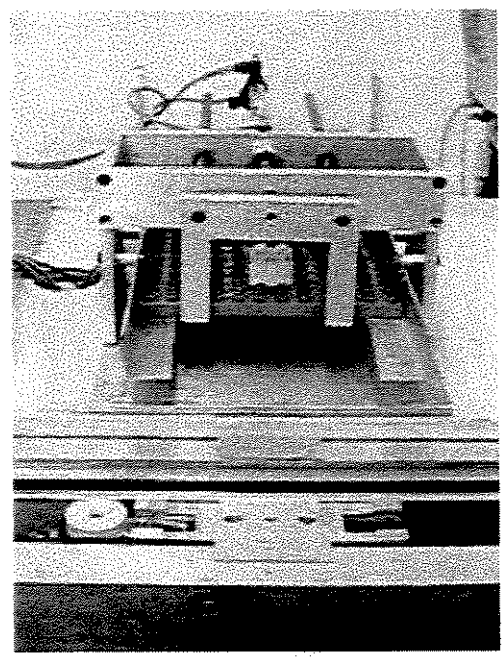


Figura 6.11 – Posto de Descarregamento

6.1.6 Sistema de Transferência

O Sistema de Transferência é composto por:

- Esteira Transportadora, dividida em 4 segmentos acionados individualmente por motores de corrente contínua
- Elevadores da Base, presentes em todos os Postos da PIPEFA. Fazem a interface física entre o Sistema de Transferência e cada um dos Postos. São acionados por cilindros pneumáticos. O Elevador da Base do Posto de Montagem/Desmontagem Lateral possui também um cilindro pneumático rotativo, para rotação da placa base em 180°. Todos estes

cilindros pneumáticos possuem sensores magnéticos de posicionamento (avançado e recuado / direita e esquerda)

- Posicionadores da Base, presentes no Leitor de Código de Barras e em todos os Postos da PIPEFA, exceto no de Carregamento. Funcionam como '*stop*' para a placa base, posicionando-a corretamente para atuação dos Elevadores ou do Leitor de Código de Barras. Também são utilizados para impedir o avanço da placa base se o Posto seguinte estiver ocupado. São acionados por pequenos cilindros pneumáticos e não possuem sensores
- Verificadores de Base, presentes no Leitor de Código de Barras, nas Curvas 2 e 4 da Esteira Transportadora e em todos os Postos da PIPEFA, exceto no de Carregamento. São sensores magnéticos que indicam a presença da placa base, sendo atuados por um pequeno disco metálico existente na parte inferior das placas bases

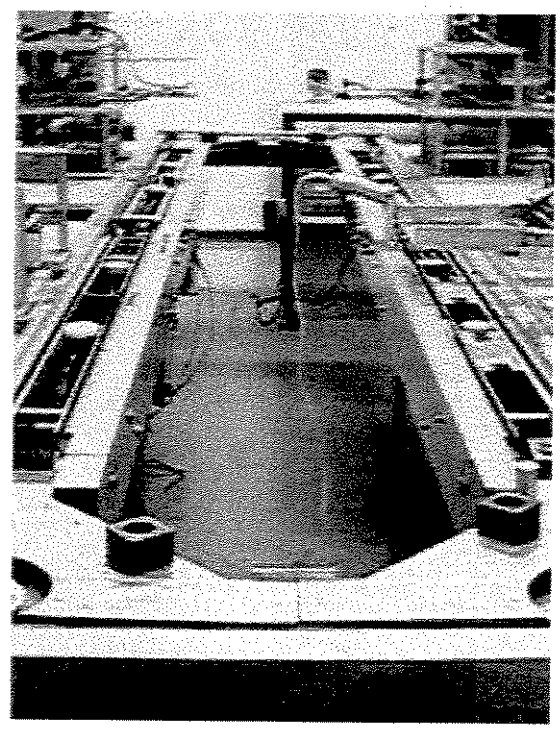


Figura 6.12 – Sistema de Transferência

Além de controlar cada um destes componentes, o Sistema de Transferência gerencia todo o fluxo da PIPEFA. Inicia a operação de cada Posto, definindo as operações que devem ser executadas. Também controla o Leitor de Código de Barras, e faz a classificação do produto

confeccionado, determinando seu destino final. A troca de informações entre o Sistema de Transferência e os Postos da PIPEFA (envio e recebimento de dados) é feita através do Sistema Supervisório Local. O Sistema de Transferência, apresentado na figura 6.12, pode ser operado manualmente através do Sistema Supervisório Local. Em Modo Automático, não permite a interferência do operador.

6.1.7 Sistema Supervisório Local

O Sistema Supervisório Local é realizado através do Software *Interact*, com finalidade principal de Interface Homem-Máquina baseada em PC.

Os recursos incorporados no Sistema Supervisório Local são:

- *Controle da PIPEFA* – permite selecionar Modo de Operação (Manual ou Automático), atuar cada Posto e o Sistema de Transferência manualmente, além de Ligar e Desligar a PIPEFA
- *Monitoramento da PIPEFA* – permite visualizar as condições gerais de cada Posto, do Leitor de Código de Barras e do Sistema de Transferência
- *Monitoramento de Alarmes* – permite visualizar as condições de alarmes ocorridos, com indicação de data, horário, operador, reconhecimento, etc.
- *Configuração de Produtos* – permite selecionar o Modo de Configuração do produto a ser confeccionado (Manual ou Leitor de Código de Barras), definir manualmente uma lista de até 10 produtos (Configuração e Quantidade) para serem confeccionados automaticamente

Atualmente, não estão incorporados recursos para Gerenciamento da Informação no Sistema Supervisório Local, uma vez que este não seria o objetivo de uma Interface Homem-Máquina. Tais recursos podem ser realizados pelo próprio *Interact* ou por outro Software/ Sistema Supervisório conectado remotamente. A estrutura atual permite implementação tanto de uma forma como de outra.

A figura 6.13 apresenta o Painel de Monitoramento da PIPEFA.

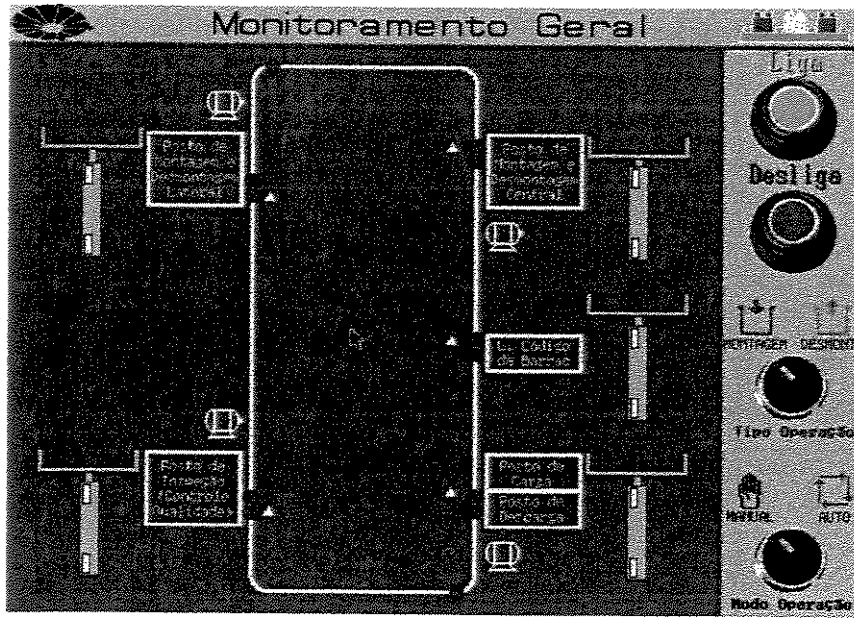


Figura 6.13 – Painel Monitoramento PIPEFA

As figuras 6.14 e 6.15 apresentam os Painéis para Configuração de Produtos e para verificação de Produtos Programados, respectivamente.

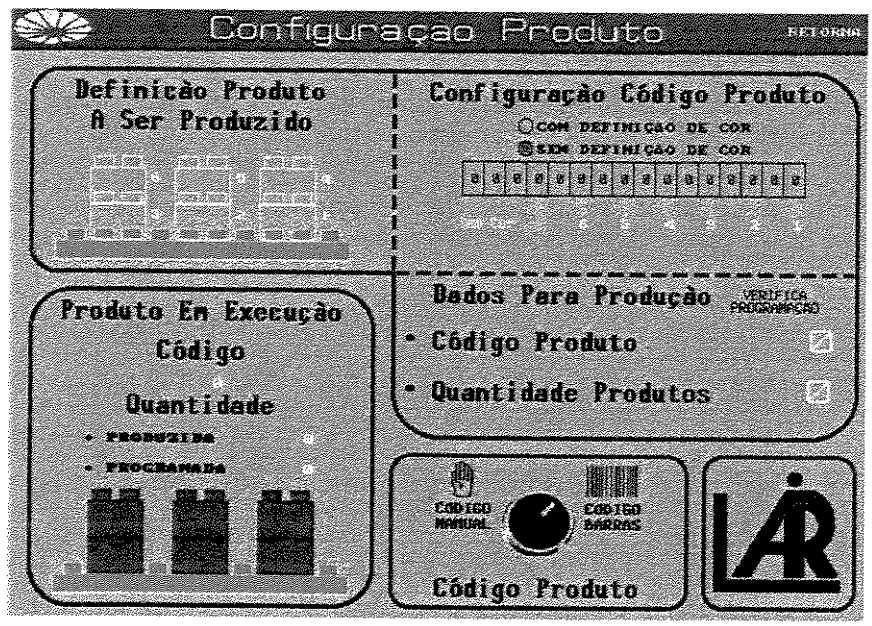


Figura 6.14 – Painel para Configuração do Produto

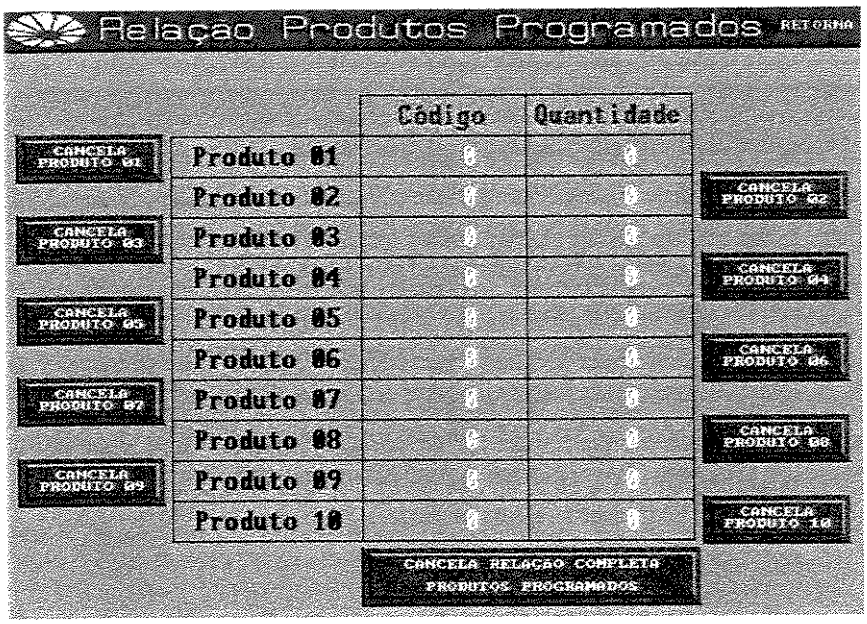


Figura 6.15 – Painel para Verificação de Produtos Programados

6.2 Implementação na PIPEFA

Os principais conceitos utilizados na implementação realizada na PIPEFA são:

1. Cada um dos Postos, bem como o Sistema de Transferência e o Leitor de Código de Barras, é uma entidade individual, independente de serem controlados pelo mesmo PLC
2. Cada uma destas entidades tem sua lógica de controle própria (Programa de Aplicação), totalmente independente das demais
3. O Sistema de Tansferência é o "coração" da PIPEFA, sendo responsável:
   - pela definição dos Postos a serem utilizados para confecção de determinado produto
   - pela iniciação e definição da operação a ser realizada por cada um dos Postos
   - pela decisão do destino de um produto confeccionado
   - pelo gerenciamento do fluxo total de produtos e informações através dos Postos
4. A comunicação (troca de informações) entre o Sistema de Transferência e cada um dos Postos da PIPEFA, incluindo o Leitor de Código de Barras, é realizada através do Sistema Supervisório (Software *Interact*, Módulo DTM – *Data Transfer Module*), ainda que sejam controlados pelo mesmo PLC
5. O Sistema de Transferência comporta-se como *Master* (Mestre) e os Postos, incluindo o Leitor de Código de Barras, como *Slaves* (Escravos)

6. Os *Slaves* executam operações pré-definidas quando iniciados adequadamente pelo *Master*. Os *Slaves* retornam ao *Master* informações necessárias ao gerenciamento do fluxo de podutos na PIPEFA (alarmes em geral, falta/excesso de produtos, etc)

A seguir são apresentados detalhes da implementação do Sistema de Transferência e do Sistema Supervisório.

6.2.4 Implementação do Sistema de Transferência

Para implementação do Sistema de Transferência foi utilizado um Controlador Lógico Programável da *PLC Direct by Koyo*, Família DL205, com a seguinte configuração:

- **1 x D2-250**.....CPU DL250, 14.8Kwords de Memória (7680 de Flash / 7168 de V-Memory), 02 Portas Seriais (RS-232 / RS-422)
- **2 x D2-16ND3-2**.....Módulo Entrada Discreta, 16 pontos 24VDC, "*sinking/sourcing*"
- **2 x D2-12TR**.....Módulo Saída Discreta, 12 pontos relé (5-30VDC, 1.5A/ponto, 3.0A/comum, 2 comuns isolados)
- **1 x D2-09B**.....Base 09 *Slots*, Fonte Interna 110/220VAC

Outros Módulos de I/O estão instalados nesta base, pois a CPU é compartilhada pelo Sistema de Transferência, pelos Postos de Carregamento, de Descarregamento e de Inspeção, e pelo Leitor de Código de Barras, conforme indicado na figura 6.3. A tabela 6.1 apresenta a disposição dos Módulos de I/O na Base, indicando o Mapeamento dos Pontos de I/O, sua Utilização e o Consumo de Corrente da Base (5VDC) e da Fonte Externa (24VDC).

SLOT	MÓDULO	MAPEAMENTO DOS PONTOS DE I/O	UTILIZAÇÃO	5VDC (mA)	24VDC (mA)
CPU	D2-250	–	–	330	–
0	D2-16ND3-2	X0–X7 / X10–X17	Sistema Transferência	100	–
1	D2-16ND3-2	X20–X27 / X30–X37	Sistema Transferência	100	–
2	D2-12TR	Y0–Y5 / Y10–Y15	Sistema Transferência	450	–
3	D2-12TR	Y20–Y25 / Y30–Y35	Sistema Transferência	450	–
4	D2-16ND3-2	X40–X47 / X50–X57	Postos Carregamento e Descarreamento	100	–
5	D2-12TR	Y40–Y45 / Y50–Y55	Posto Carregamento e Descarreamento	450	–
6	D2-16ND3-2	X60–X67 / X70–X77	Posto Inspeção	100	–
7	D2-12TR	Y60–Y65 / Y70–Y75	Posto Inspeção	450	–
			TOTAL	**2.530**	**0**
			Potência Fornecida Pela Base D2-09B	**2.600**	**300**

Tabela 6.1 – Configuração do PLC Utilizado e Cálculo do Consumo de Corrente

Como visto, esta configuração não apresenta limitações quanto ao consumo de corrente, uma vez que o consumo total dos Módulos de I/O e da CPU é menor que o máximo fornecido pela Base. Os Módulos de I/O utilizados não necessitam de alimentação externa (24VDC), porém é utilizada uma Fonte Externa para alimentação dos Dispositivos de Entrada e de Saída.

A figura 6.16 apresenta com detalhes os elementos utilizados na implementação do Sistema de Transferência. Foram mantidas as referências (*nicknames*) utilizadas no Programa de Aplicação, sendo que as apresentadas em azul (caracteres minúsculos) são dispositivos de entrada (sensores) e estão conectados aos Módulos D2-16ND3-2, e as apresentadas em vermelho (caracteres maiúsculos) são dispositivos de saída (atuadores) e estão conectados aos Módulos D2-12TR.

O Anexo III apresenta, em detalhes, o Mapeamento dos Pontos de I/O utilizados pelo Sistema de Transferência, e o Anexo IV os Esquemas Elétricos correspondentes.

A CPU DL250 possui duas portas de comunicação, sendo que a Porta 1 foi utilizada para conexão ao Sistema Supervisório Local e a Porta 2 para conexão ao Leitor de Código de Barras, pois somente esta permite a configuração '*Non-Sequence*' (Seqüência Não-Definida). A tabela 6.2 apresenta as características das portas seriais disponíveis na CPU utilizada.

	Porta 1 (Superior)	Porta 2 (Inferior)
Conector	Jack RJ12 (Fêmea, 6 Pinos)	DB-15 (VGA, Fêmea)
Protocolos	Propietário, *Direct*Net (*slave*)	Propietário, *Direct*Net (*master/slave*), MODBUS RTU (*master/slave*), I/O Remoto, *Non-Sequence*
Padrão	RS-232-C (9.600bps)	RS-232-C ou RS-422 (até 38.400 bps)
Endereço	Configurável (1-90)	Configurável (1-90)

Tabela 6.2 – Características das Portas de Comunicação (CPU DL250)

O Programa de Aplicação desenvolvido em Linguagem Ladder (instruções RLL^{PLUS}), através do Software de Programação da *PLC Direct by Koyo* (*Direct*SOFT) encontra-se à disposição no Laboratório de Automação Integrada e Robótica da UNICAMP.

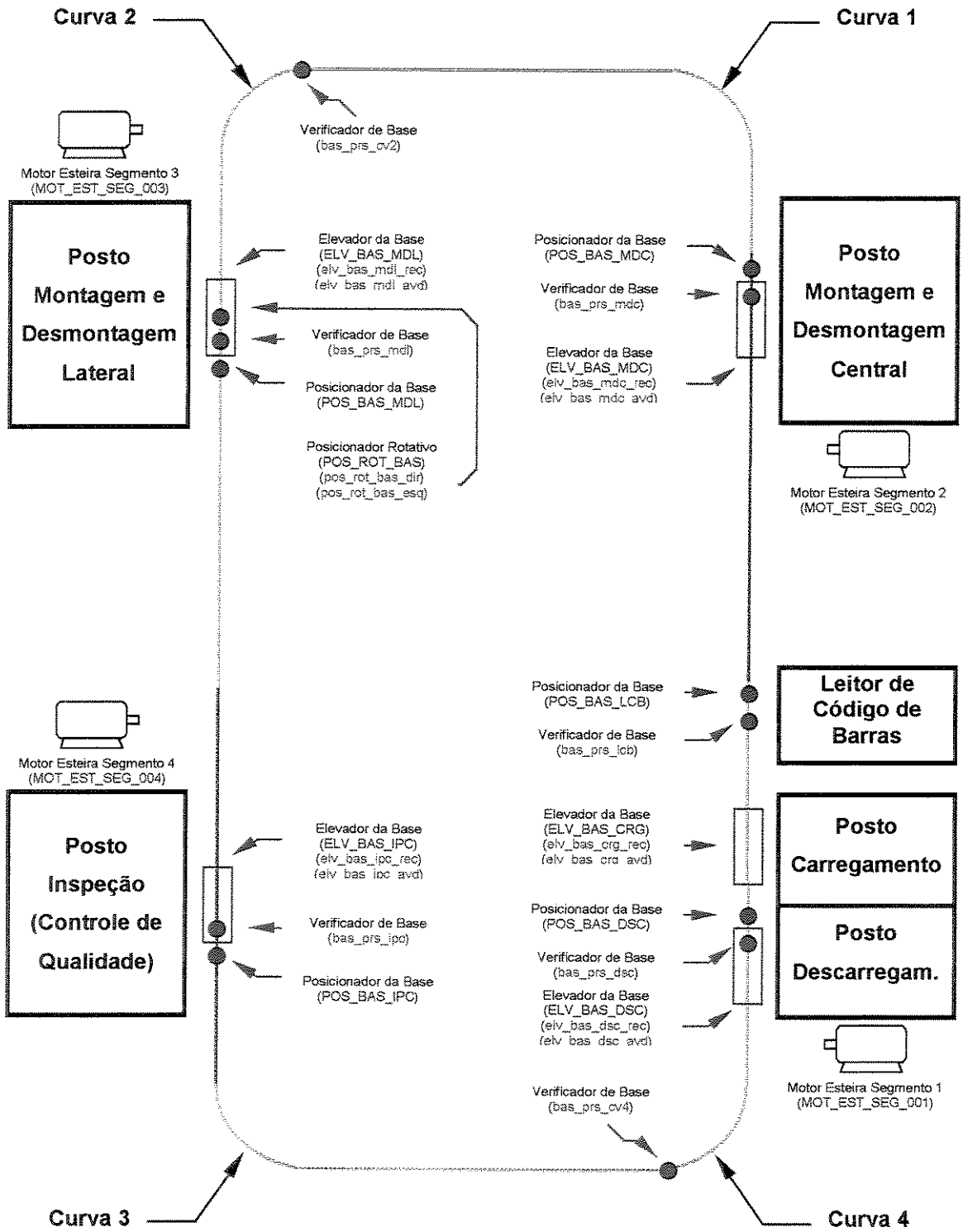


Figura 6.16 – Implementação do Sistema de Transferência

A implementação foi baseada na descrição por Diagrama Funcional Seqüencial (*SFC*). O Anexo V apresenta o *SFC* para operação do Sistema de Transferência em Modo Automático.

6.2.2 Implementação do Sistema Supervisório

Os Módulos utilizados na configuração do Software *Interact* para implementação da PIPEFA são:

- **APM (*Application Manager*)** – Gerenciador Central, responsável pelo gerenciamento total da Aplicação
- **PTM (*Panel ToolKit Module*)** – Kit de Ferramentas do Painel, utilizado para criação dos Painéis de Controle e Monitoramento, através das Ferramentas de Escrita e Leitura
- **GMM (*Graphic Monitoring Module*)** – Animação Gráfica, utilizado para criação da área gráfica (animada e estática) dos Painéis de Monitoramento e Configuração
- **AMM (*Alarm Management Module*)** – Gerenciador de Alarmes, utilizado para criação do Sistema de Monitoramento de Alarmes
- **DTM (*Data Transfer Module*)** – Transferência de Dados, utilizado para troca de dados entre o Sistema de Transferência e os demais Postos da PIPEFA, além do Leitor de Código de Barras

Além dos Módulos, fazem parte desta configuração do *Interact* os seguintes Drivers:

- **TICM (*Texas Instruments Communication Module*[3])** – Driver para comunicação com PLCs *Koyo*, configurado de acordo com a porta de comunicação do PLC
- **DVR (*Demo Driver*)** – Driver demonstrativo que opera apenas internamente ao PC, não permitindo Escrita ou Leitura em qualquer tipo de dispositivo externo. É utilizado para realização de lógica interna ao Sistema Supervisório desenvolvido

Cada um dos Módulos utilizados nesta Aplicação tem características próprias, de acordo com sua finalidade, conforme apresentado no Capítulo 5.

---

[3] O nome TICM deve-se ao fato de que a Koyo (Japão) forneceu algumas Famílias de PLCs para as empresas Texas Instruments, Siemens e GE, durante vários anos. Ao surgir a *PLC Direct by Koyo* (Janeiro '94 – Estados Unidos), o *Interact* já possuía este Driver, não sendo realizada qualquer alteração de nomenclatura

A forma de configuração dos itens de cada Módulo é idêntica, ou seja, utiliza-se uma sintaxe padrão independente da função a ser realizada. A figura 6.17 apresenta a sintaxe utilizada para comunicação através do Driver TICM.

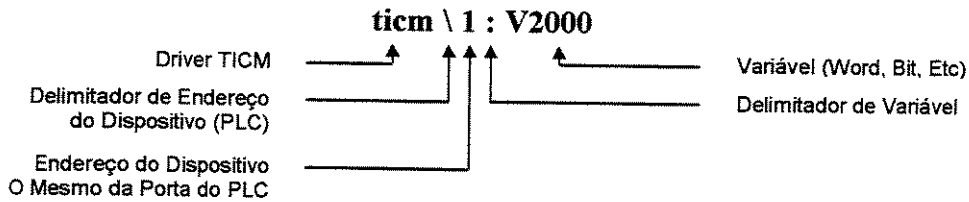


Figura 6.17 – Sintaxe Padrão para Comunicação com PLCs Koyo (Driver TICM)

O Anexo VI apresenta algumas orientações quanto à execução do Software *Interact*.

Os arquivos da Aplicação desenvolvida para este Sistema Supervisório também encontram-se à disposição no Laboratório de Automação Integrada e Robótica da UNICAMP.

Neste capítulo foi apresentada a estrutura completa da PIPEFA, com ênfase à constituição e operação do Sistema de Tansferência, sendo finalizado com a descrição da implementação através de PLC e Sistema Supervisório Local.

Foram realizados testes que comprovaram a eficiência dos elementos utilizados na estruturação e implementação deste sistema.

Capítulo 7

Conclusões e Perspectivas Futuras

Neste trabalho foram abordados os aspectos referentes à descrição e implementação de SAPs, através da apresentação de norma para descrição de sistemas seqüenciais (*SFC – Sequential Function Chart*), apresentação de equipamento industrial para controle (PLC) e sua programação, e apresentação de Sistemas Supervisórios, visando a integração de sistemas através de um modelo consistente que garanta conectividade e flexibilidade.

Dentro desse objetivo foram apresentados didaticamente, nos capítulos iniciais, ferramentas para descrição e implementação de SAPs, levando-se em consideração normas existentes e equipamentos industriais (hardware e software), proporcionando a elaboração de uma referência bilbiográfica para utilização acadêmica e industrial.

A validação desses conceitos foi apresentada no capítulo final, utilizando a PIPEFA, através de implementação de um SAP composto por Postos Operacionais controlados por PLC e Sistema Supervisório.

Finalmente, os seguintes aspectos podem ser ressaltados:

1. A elaboração dessa dissertação de mestrado foi realizada procurando proporcionar ao leitor uma referência bibliográfica atualizada dos assuntos abordados, tanto a nível acadêmico como industrial

2. Os resultados obtidos mostram que a utilização de ferramentas para estruturação de SAPs proporcionam uma implementação consistente, permitindo atualizações futuras de forma rápida e fácil
3. Os conceitos abordados são genéricos, podendo ser aplicados a outros SAPs tanto a nível de estruturação (*SFC*) como de implementação (PLC e Sistema Supervisório)

Como perspectivas futuras, para novos trabalhos, podemos destacar:

1. Com relação à estruturação

- Integração de Programas para Gestão da Produção
- Sistemas de Informação envolvendo redes de comunicação entre os diferentes níveis de um Sistema Industrial

2. Com relação à implementação

- Controle baseado em PC
- Equipamentos (hardware e software) que atendam à norma IEC-1131

Além destas, outras evoluções que poderiam ser implementadas na PIPEFA, objetivo de nosso estudo

- utlização de sensores de cor
- codificação dos produtos por 'etiquetas magnéticas' ou eletronicamente
- acesso remoto, por rede local e modem
- alimentação automática dos Postos Operacionais por manipuladores industriais (robôs)

Referências Bibliográficas

Bryan, L.A., Bryan, E.A. *Programmable Controllers - Workbook and Study Guide*. Atlanta: Industrial Text, 1988, 327p.

Celati, C. *Automazione Industriale*. Milano: Editore Ulrico Hoepli Milano, 1995, 619p.

Chiacchio, P. *PLC e Automazione Industriale*. Milano: McGraw-Hill Libri Italia, 1996, 220p.

CTC – Computer Technology Corporation. *Interact 5 Student Training Manual*. Cincinnati, 1997.

IEC – International Eletrotechnical Commission, Genebra. IEC-60848; *Preparation of Function Charts for Control Systems*. Genebra, 1988, 99p.

Martins, M.P. *Estruturação da Parte de Comando de um Sistema Automatizado de Produção com Ênfase na Implementação de um Sistema de Supervisão*. Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1998. 158p. Tese (Doutorado)

PLC Direct by Koyo. *DL205 Analog I/O Manual*. 3.ed. Atlanta, 1997.

PLC Direct by Koyo. *DL205 User Manual*.2.ed. Atlanta, 1997.

PLC Direct by Koyo. *DL405 Analog I/O Manual*. 2.ed., rev.A. Atlanta, 1998

PLC Direct by Koyo. *DL405 User Manual.* 3.ed. Atlanta, 1997.

Rosário, J.M., Frachet, J.P. *Métodos e Ferramentas em Automação Industrial e Produção para o Desenvolvimento da Qualidade e da Produtividade nas PME/PMI.* Projeto de Pesquisa, CAPES/COFECUB, Junho,1996.

Roussel, J.M. *Editorial.* Automatique Productique Informatique Industrielle, v.27-1, 1993.

Stenerson, J. *Fundamentals of Programmable Logic Controllers, Sensors, and Communications.* 2.ed. New Jersey: Regents/Prentice Hall, 1999, 562p.

Anexo I

Memórias Semicondutoras

Memórias são dispositivos responsáveis pelo armazenamento das informações (dados) de um sistema. Os principais tipos de memórias disponíveis são apresentadas na tabela A1.1.

Tipos de memórias	Aplicação / Característica
Semicondutor (Bipolar, MOS, etc)	Microprocessador/ Buffer de Dados/ Rápida
Magnética (disco ou fita)	Armazenamento /Alta Capacidade / Lenta
Óptica	Armazenamento/ Alta Capacidade / Veloz

Tabela A1.1 – Tipos de Memória

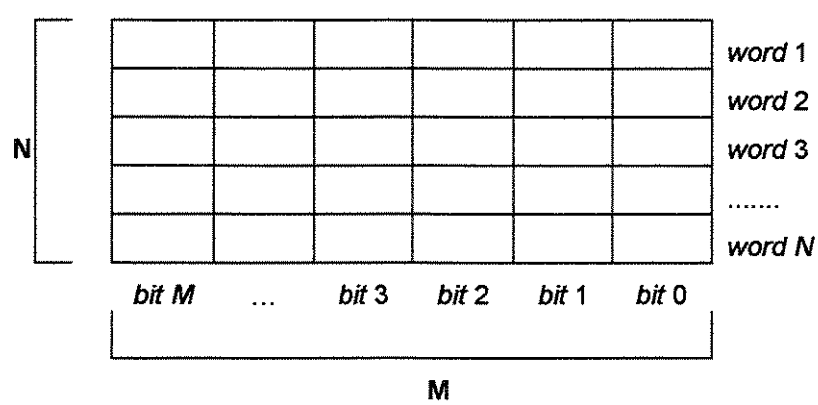


Figura A1.1 – Organização de uma Memória Semicondutora

As memórias semicondutoras, objetivo desta revisão, são constituídas por "células" de armazenamento de 1 *bit* (por exemplo, Flip-Flop) organizadas em matrizes. As matrizes contem **N** linhas por **M** colunas. (**N** *words* de **M** *bits*), conforme apresentado na figura A1.1.

Basicamente, as memórias semicondutoras são classificadas em dois tipos:

- Memórias de Escrita e Leitura (*RAM - Random Access Memory* – Memória de Acesso Aleatório)
- Memórias Apenas de Leitura (*ROM - Read Only Memory*)

Memórias *RAM*

As Memórias *RAM* são utilizadas para armazenamento de dados temporários. Tratam-se de memórias voláteis, isto é, perdem os dados armazenados na falta de alimentação.

Existem dois tipos básicos de tecnologia para as Memórias *RAM*:

- *RAM Estática (SRAM – Static RAM)* – o elemento de armazenamento é implementado por Flip-Flops convencionais. Estas memórias são rápidas, apresentando alto consumo e alta confiabilidade
- *RAM Dinâmica (DRAM – Dynamic RAM)* – o elemento de armazenamento é implementado por capacitores. São razoavelmente rápidas, apresentando baixo consumo, menor confiabilidade e maior capacidade de armazenamento por área

Memórias *ROM*

Tratam-se de memórias não-voláteis, isto é, mantêm os dados armazenados mesmo na falta de alimentação. Os tipos normalmente disponíveis:

- *Mask ROM* (*ROM Máscara*) – são memórias fabricadas sob encomenda, usando a 'máscara' fornecida pelo cliente. Os bits já vem em 0 ou 1, e não podem ser alterados
- *PROM* (*Programmable ROM* – ROM Programável) – possibita a gravação dos bits por parte do usuário. Porém, uma vez gravada não é possível ser alterada
- *EPROM* (*Erasable Programmable ROM* – ROM Programável e Apagável) – possibilita a gravação e alteração dos bits por parte do usuário. Para isto, é necessário realizar o apagamento total da memória através de luz ultra-violeta

- *EEPROM/ E^2PROM (Electrically Erasable Programmable ROM* – ROM Programável e Apagável Eletricamente) – possibilita a gravação e alteração dos bits por parte do usuário. A alteração é feita eletricamente (por níveis de tensão), sem necessidade de remover a memória do circuito
- *FLASH EPROM* – semelhante à EEPROM, porém com tecnologia que possibilita maior número de ciclos de alteração e maior facilidade de operação

Anexo II

Código de Barras

O formato do código de barras presente nas placas bases e utilizado para definição do produto a ser confeccionado pela PIPEFA é apresentado na figura A2.1.

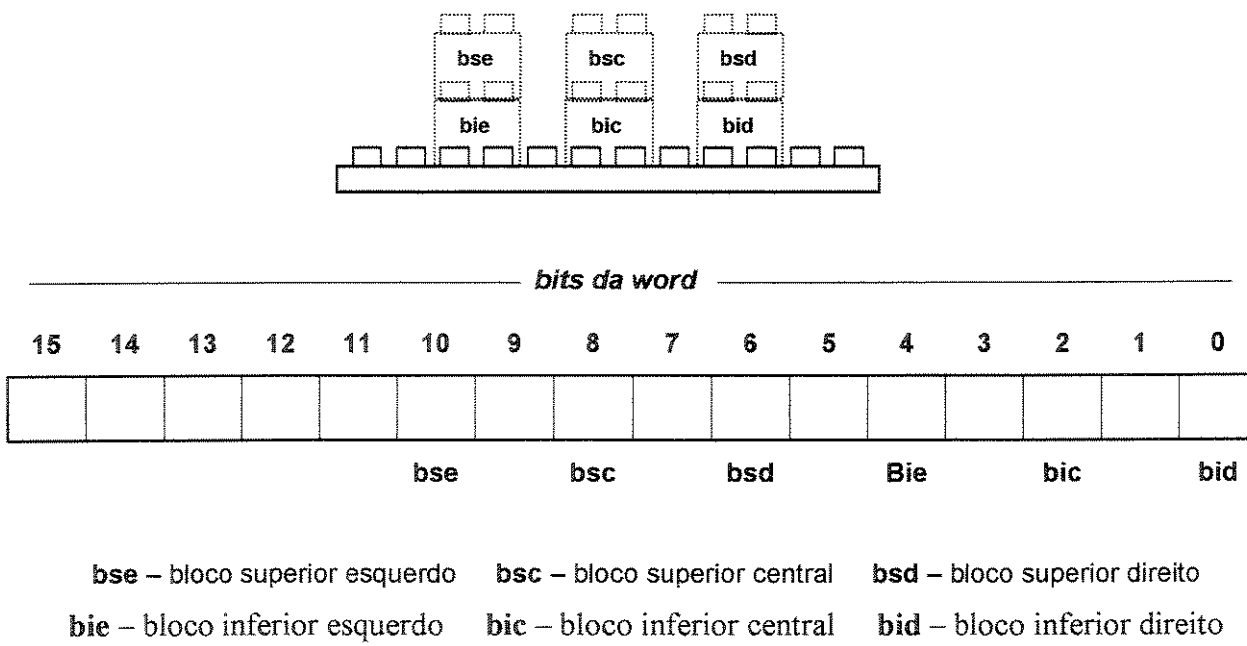


Figura A2.1 – Formato Código Barras

Como pode ser observado, cada bloco consome apenas um *bit* da *word* utilizada para definição de código do produto. Se este *bit* for igual a zero (0), indica ausência de cubo na

posição correspondida. Caso contrário, *bit* igual a um (1), indica presença de cubo, definindo que o mesmo deve ser montado pelo respectivo Posto de Montagem/Desmontagem.

Os produtos possíveis de serem confeccionados atualmente pela PIPEFA, e seus respectivos códigos, são apresentados na tabela A2.1.

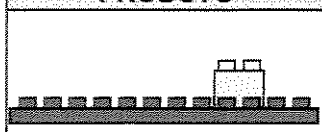
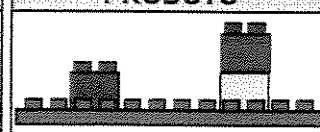
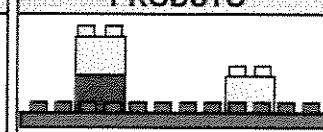
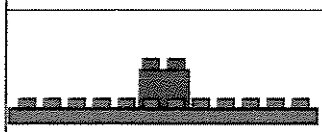
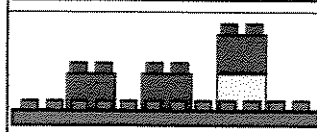
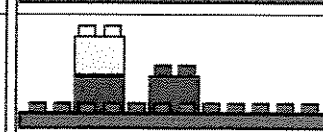
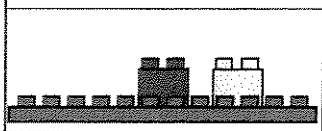
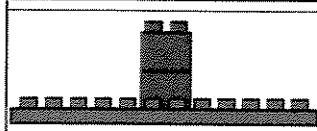
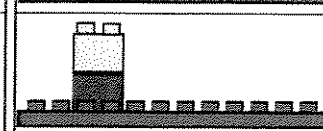
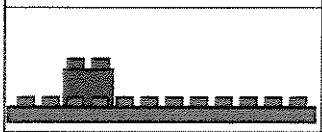
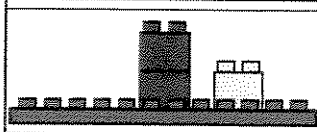
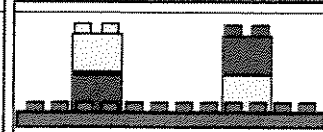
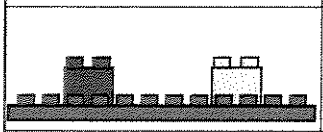
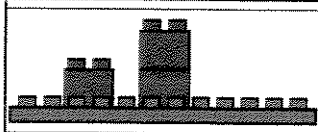
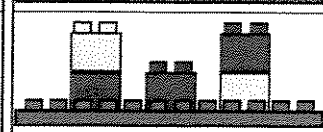
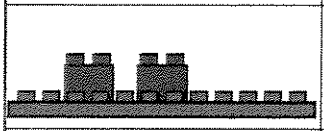
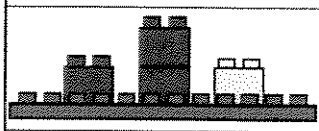
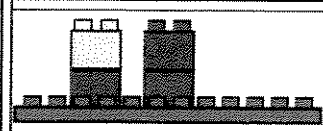
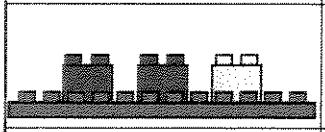
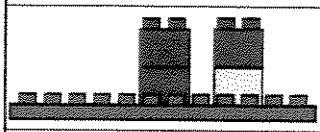
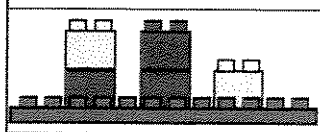
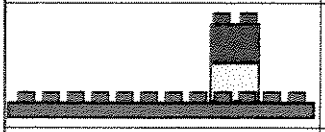
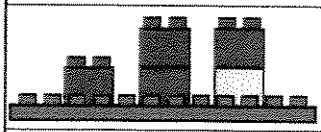
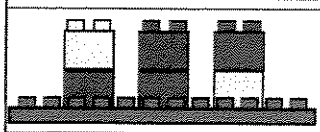
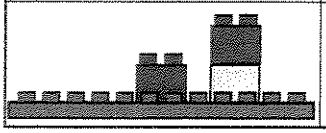
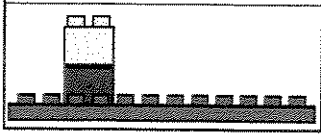
PRODUTO	COD	PRODUTO	COD	PRODUTO	COD
	0001		0081		1041
	0004		0085		1044
	0005		0260		1045
	0016		0261		1105
	0017		0276		1109
	0020		0277		1300
	0021		0325		1301
	0065		0341		1365
	0069		1040		

Tabela A2.1 – Relação Produtos PIPEFA

Anexo III

Mapeamento dos Pontos de I/O (Sistema de Transferência)

A seguir é apresentado o mapeamento completo dos Pontos de I/O, referente à implementação realizada na PIPEFA.

São apresentados os elementos (com breve descritivo), os Dispositivos de Entrada e de Saída relacionados e a função de cada um deles, o *nickname* (apelido, referência) utilizado na documentação do Programa de Aplicação (Linguagem Ladder), os endereços do PLC (Entrada e Saída), e a identificação na régua de borne.

SISTEMA DE TRANSFERÊNCIA

Relação de Entradas e Saídas

Elemento	Dispositivo de Entrada	Dispositivo de Saída	Função	Nickname	Ref. PLC Entrada	Ref. PLC Saída	Régua Borne
Motor Esteira Transportadora Segmento 1		Relé 12 VDC	Avanço da Esteira	MOT_EST_SEG_001		Y20	25
Motor Esteira Transportadora Segmento 2		Relé 12 VDC	Avanço da Esteira	MOT_EST_SEG_002		Y21	28
Motor Esteira Transportadora Segmento 3		Relé 12 VDC	Avanço da Esteira	MOT_EST_SEG_003		Y22	32
Motor Esteira Transportadora Segmento 4		Relé 12 VDC	Avanço da Esteira	MOT_EST_SEG_004		Y23	36
Cilindro Pneumático do Elevador da Base (Posto Carregamento)	Sensor Magnético 24 VDC Sensor Magnético 24 VDC	Solenóide 24 VDC	Avanço do Cilindro Cilindro Recuado Cilindro Avançado	ELV_BAS_CRG elv_bas_crg_rec elv_bas_crg_avd	 X0 X1	Y0	23 5 6
Cilindro Pneumático do Posicionador da Base (Leitura do Código de Barras)		Solenóide 24 VDC	Avanço do Cilindro	POS_BAS_LCB		Y1	24
Verificador de Base para Leitura do Código de Barras	Sensor Capacitivo 24 VDC		Base Presente	bas_prs_lcb	X2		7
Cilindro Pneumático do Posicionador da Base (Posto Mont./Desm. Central)		Solenóide 24 VDC	Avanço do Cilindro	POS_BAS_MDC		Y2	26
Verificador de Base no Posto Mont./Desm. Central	Sensor Capacitivo 24 VDC		Base Presente	bas_prs_mdc	X3		8
Cilindro Pneumático do Elevador da Base (Posto Mont./Desm. Central)	Sensor Magnético 24 VDC Sensor Magnético 24 VDC	Solenóide 24 VDC	Avanço do Cilindro Cilindro Recuado Cilindro Avançado	ELV_BAS_MDC elv_bas_mdc_rec elv_bas_mdc_avd	 X4 X5	Y3	27 9 10
Cilindro Pneumático do Posicionador da Base (Posto Mont./Desm. Lateral)		Solenóide 24 VDC	Avanço do Cilindro	POS_BAS_MDL		Y4	30
Verificador de Base no Posto Mont./Desm. Lateral	Sensor Capacitivo 24 VDC		Base Presente	bas_prs_mdl	X6		12

SISTEMA DE TRANSFERÊNCIA

***Relação de Entradas e Saídas* (Continuação)**

Elemento	Dispositivo de Entrada	Dispositivo de Saída	Função	Nickname	Ref. PLC		Régua
					Entrada	Saída	Borne
Cilindro Pneumático do		Solenóide 24 VDC	Avanço do Cilindro	ELV_BAS_MDL		Y5	29
Elevador da Base	Sensor Magnético 24 VDC		Cilindro Recuado	elv_bas_mdl_rec	X7		13
(Posto Mont./Desm. Lateral)	Sensor Magnético 24 VDC		Cilindro Avançado	elv_bas_mdl_avd	X10		14
Cilindro Pneumático		Solenóide 24 VDC	Gira Cilindro à Direita	POS_GIR_BAS		Y10	31
Giratório da Base	Sensor Magnético 24 VDC		Cilindro à Direita	pos_gir_bas_dir	X11		16
(Posto Mont./Desm. Lateral)	Sensor Magnético 24 VDC		Cilindro à Esquerda	pos_gir_bas_esq	X12		15
Cilindro Pneumático do Posicionador da Base (Posto Inspeção)		Solenóide 24 VDC	Avanço do Cilindro	POS_BAS_IPC		Y11	34
Verificador de Base no Posto Inspeção	Sensor Capacitivo 24 VDC		Base Presente	bas_prs_ipc	X13		17
Cilindro Pneumático do		Solenóide 24 VDC	Avanço do Cilindro	ELV_BAS_IPC		Y12	35
Elevador da Base	Sensor Magnético 24 VDC		Cilindro Recuado	elv_bas_ipc_rec	X14		19
(Posto Inspeção)	Sensor Magnético 24 VDC		Cilindro Avançado	elv_bas_ipc_avd	X15		20
Cilindro Pneumático do Posicionador da Base (Posto Descarregamento)		Solenóide 24 VDC	Avanço do Cilindro	POS_BAS_DSC		Y13	21
Verificador de Base no Posto Descarregamento	Sensor Capacitivo 24 VDC		Base Presente	bas_prs_dsc	X16		2
Cilindro Pneumático do		Solenóide 24 VDC	Avanço do Cilindro	ELV_BAS_DSC		Y14	22
Elevador da Base	Sensor Magnético 24 VDC		Cilindro Recuado	elv_bas_dsc_rec	X17		3
(Posto Descarregamento)	Sensor Magnético 24 VDC		Cilindro Avançado	elv_bas_dsc_avd	X20		4
Verificador de Base (Antes da Curva 2)	Sensor Capacitivo 24 VDC		Base Presente	bas_prs_cv2	X21		11
Verificador de Base (Antes da Curva 4)	Sensor Capacitivo 24 VDC		Base Presente	bas_prs_cv4	X22		1
Emergência Manual	Botoeira com Trava (NF)		Emergência Acionada	emg_man	27		XX

Obs.:

'XX' = Elemento não instalado na versão atual

Anexo IV

Esquemas Elétricos (Sistema de Transferência)

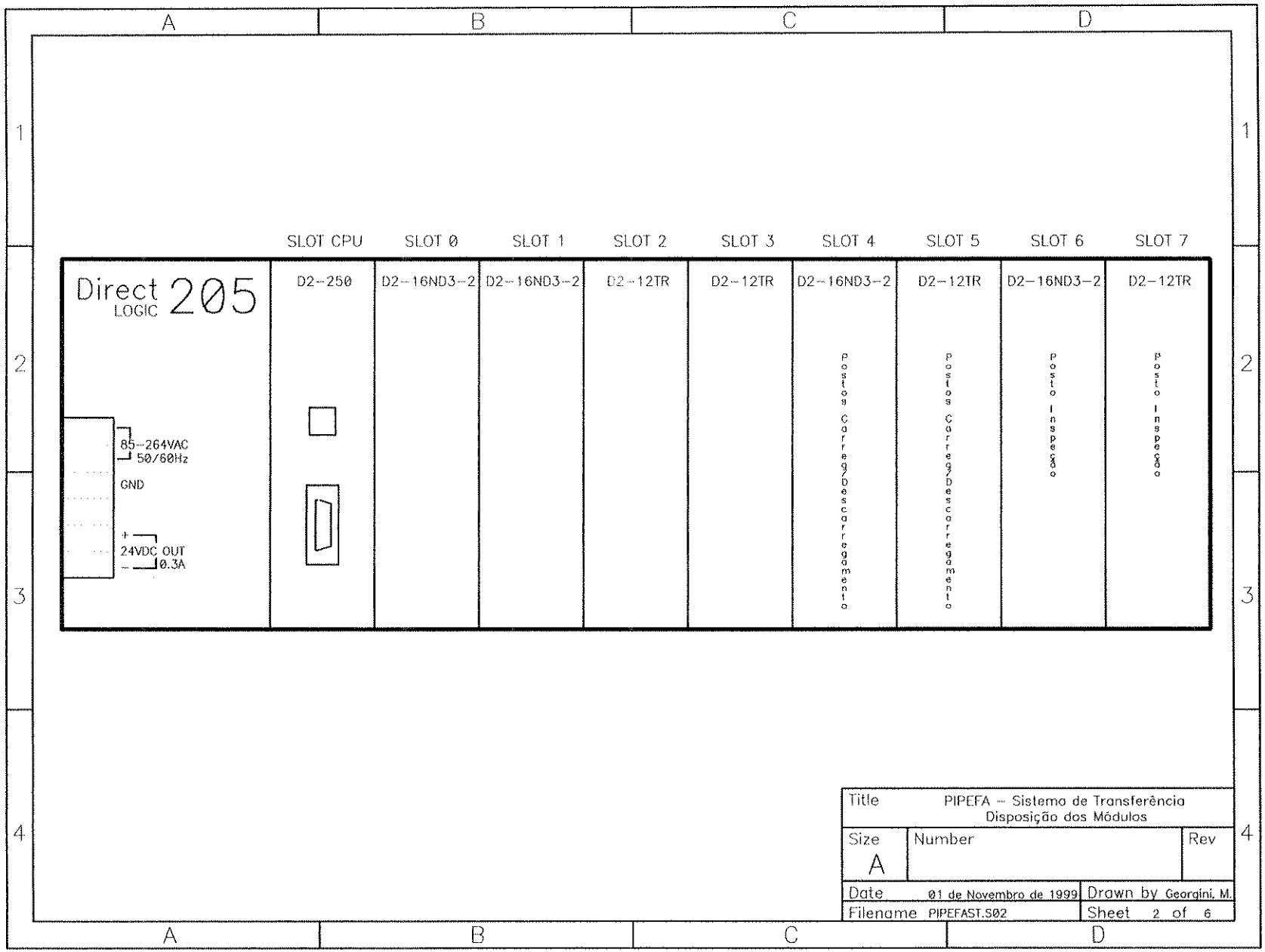
A seguir são apresentados os esquemas elétricos relacionados ao Sistema de Transferência.

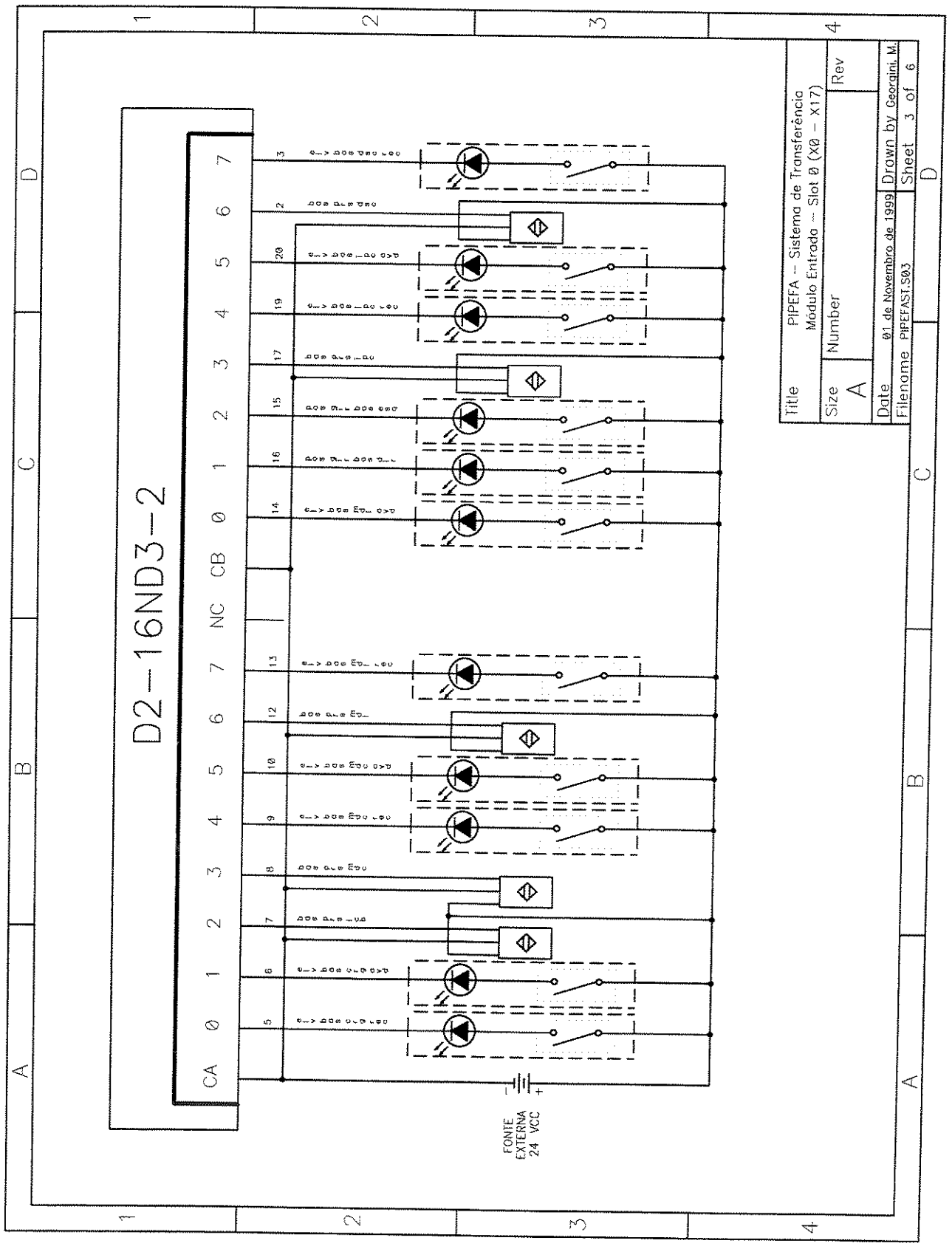
PIPEFA

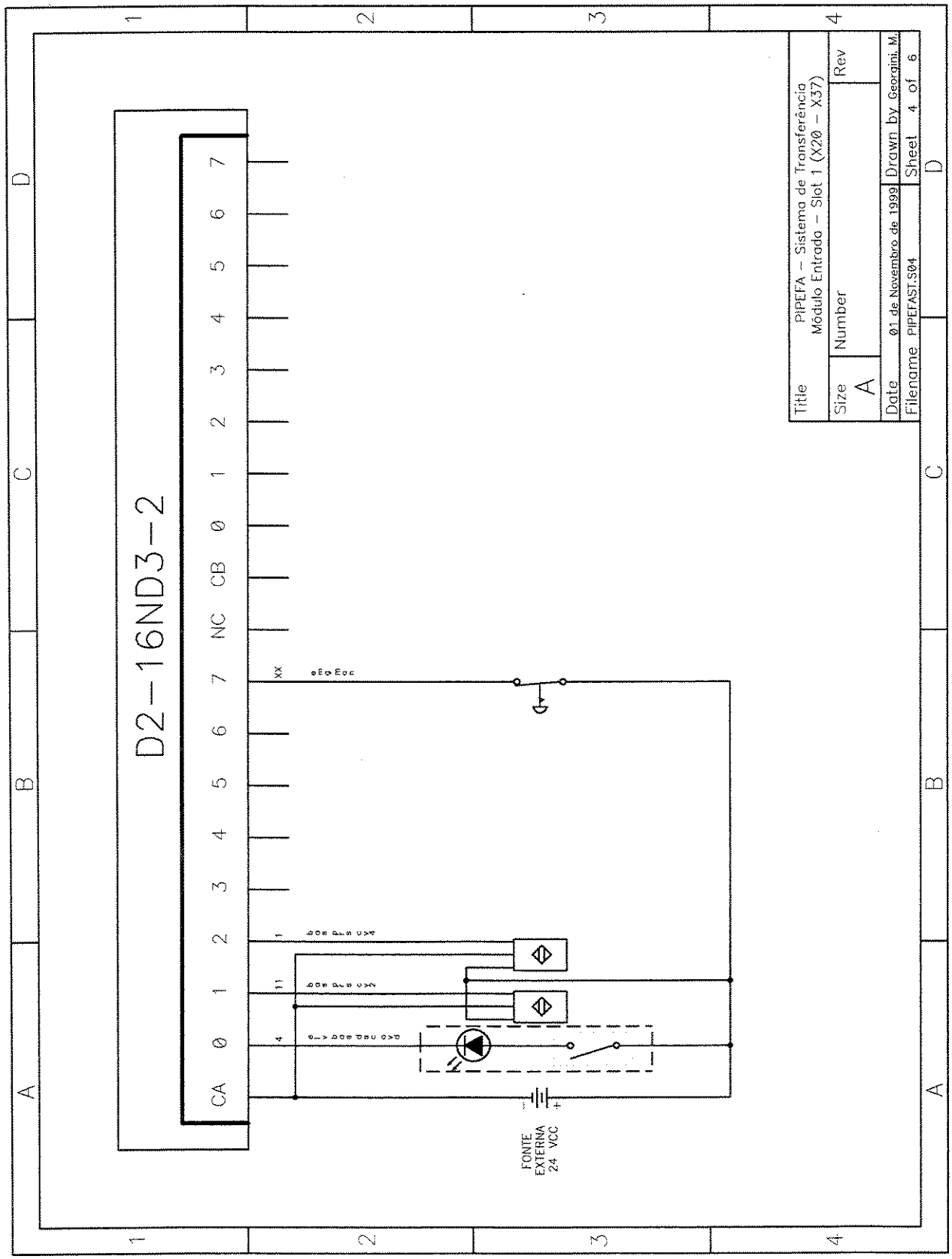
Sistema de Transferência

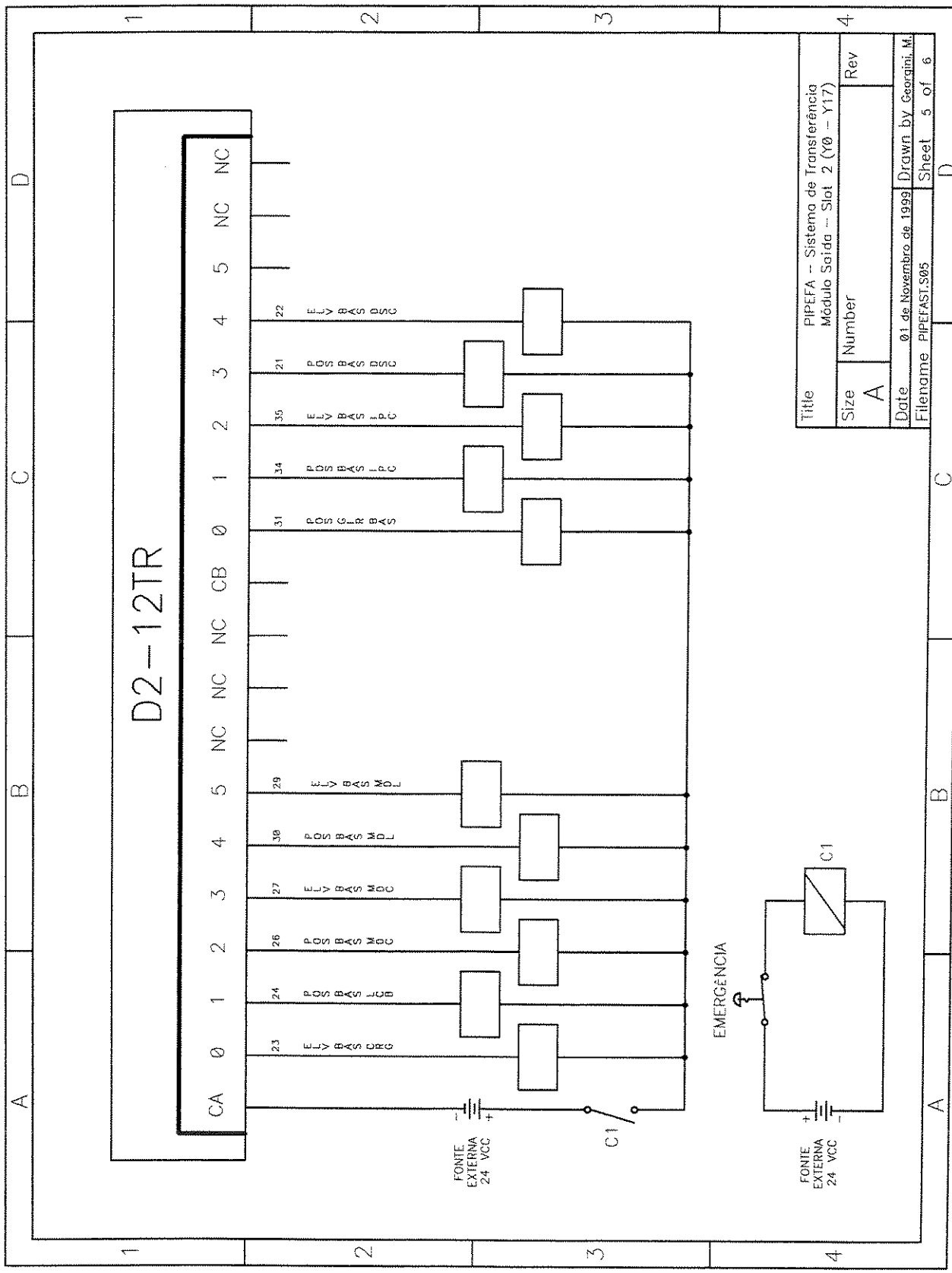
— — ESQUEMAS ELÉTRICOS — —

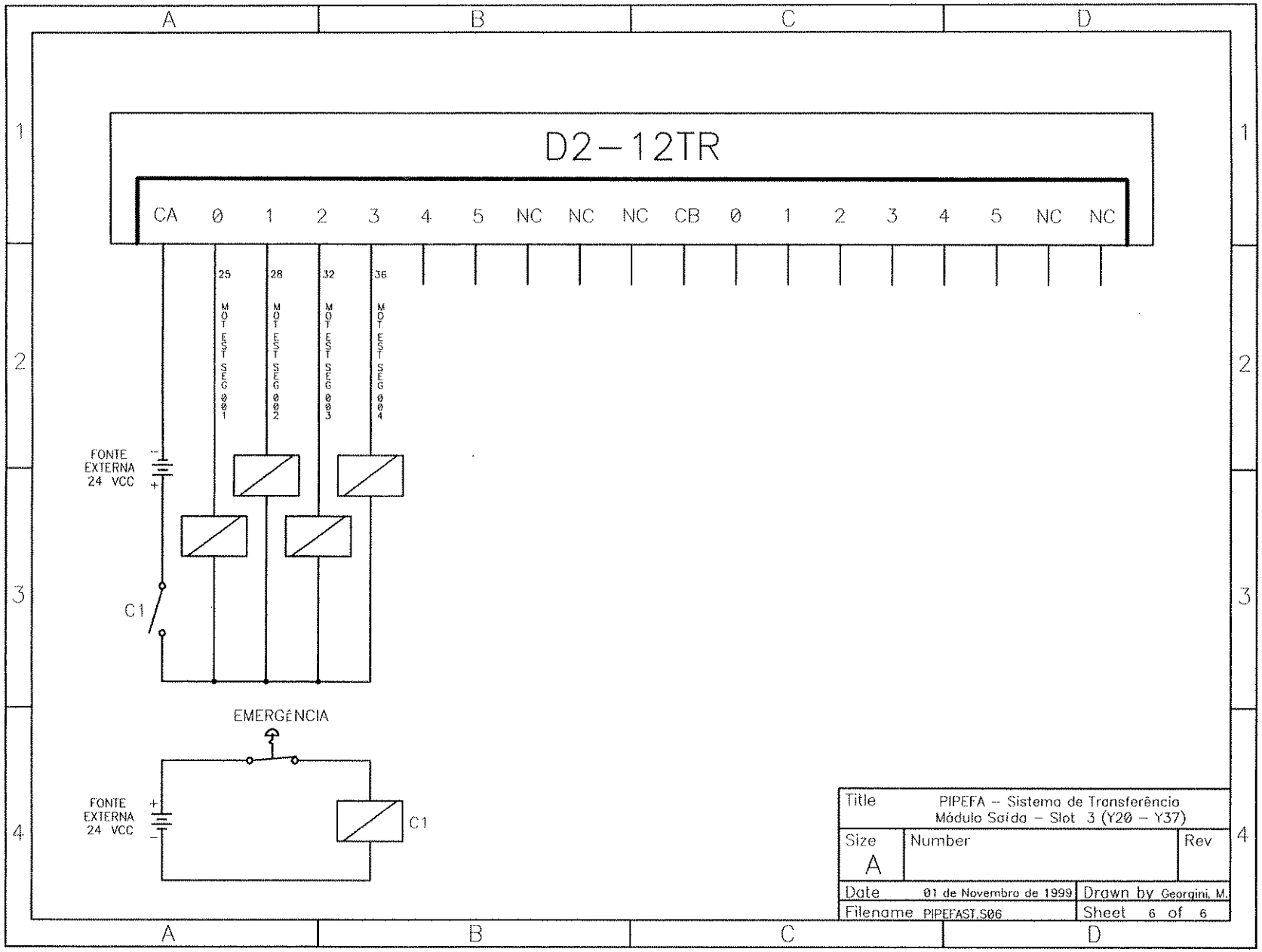
Title	PIPEFA -- Sistema de Transferência Esquemas Elétricos		
Size A	Number		Rev
Date	01 de Novembro de 1999	Drawn by	Georgini, M.
Filename	PIPEFAST.S01	Sheet	1 of 6


A
B
C
D
1
2
3
4
SLOT CPU
SLOT 0
SLOT 1
SLOT 2
SLOT 3
SLOT 4
SLOT 5
SLOT 6
SLOT 7
Direct 205
LOGIC
D2-250
D2-16ND3-2
D2-16ND3-2
D2-12TR
D2-12TR
D2-16ND3-2
D2-12TR
D2-16ND3-2
D2-12TR
Posto Carregamento
Posto Descarregamento
Posto Inspeção
Posto Inspeção
85-264VAC
50/60Hz
GND
+
24VDC OUT
-
0.3A
Title
PIPEFA -- Sistema de Transferência
Disposição dos Módulos
Size
A
Number
Rev
Date
01 de Novembro de 1999
Drawn by Georgini, M.
Filename PIPEFAST.S02
Sheet 2 of 6


D2-16ND3-2
CA 0 1 2 3 4 5 6 7 NC CB 0 1 2 3 4 5 6 7
FONTE EXTERNA 24 VCC
Title PIPEFA -- Sistema de Transferência Módulo Entrada -- Slot 0 (X0 - X17)
Size A
Number
Rev
Date 01 de Novembro de 1999
Drawn by Georgini, M.
Filename PIPEFAST.S03
Sheet 3 of 6


D2-16ND3-2
CA 0 1 2 3 4 5 6 7 NC CB 0 1 2 3 4 5 6 7
FONTE EXTERNA 24 VCC
Title PIPEFA – Sistema de Transferência Módulo Entrada – Slot 1 (X20 – X37)
Size A
Number
Rev
Date 01 de Novembro de 1999
Drawn by Georgini, M.
Filename PIPEFAST.S04
Sheet 4 of 6

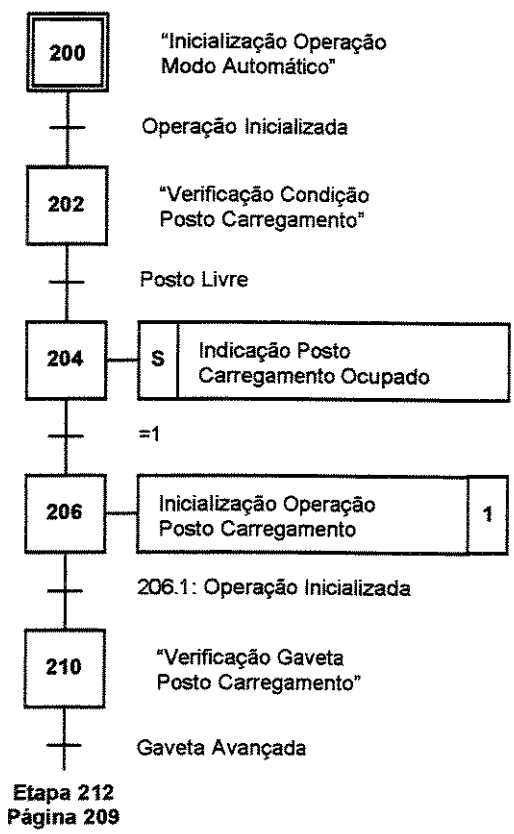

D2-12TR
CA 0 1 2 3 4 5 NC NC NC CB 0 1 2 3 4 5 NC NC
23 24 26 27 30 29 31 34 35 21 22
EJV BAS SBC
POS BAS LCB
POS BAS MOC
EJV BAS MOC
POS BAS MOD
EJV BAS MOD
POS SLR BAS
POS BAS LPC
EJV BAS LPC
POS BAS BDSC
EJV BAS BDSC
FONTE EXTERNA 24 VCC
C1
EMERGÊNCIA
C1
FONTE EXTERNA 24 VCC
Title PIPEFA - Sistema de Transferência Módulo Saída - Slot 2 (Y0 - Y17)
Size A
Number
Rev
Date 01 de Novembro de 1999
Drawn by Georgini, M.
Filename PIPEFAST.S05
Sheet 5 of 6

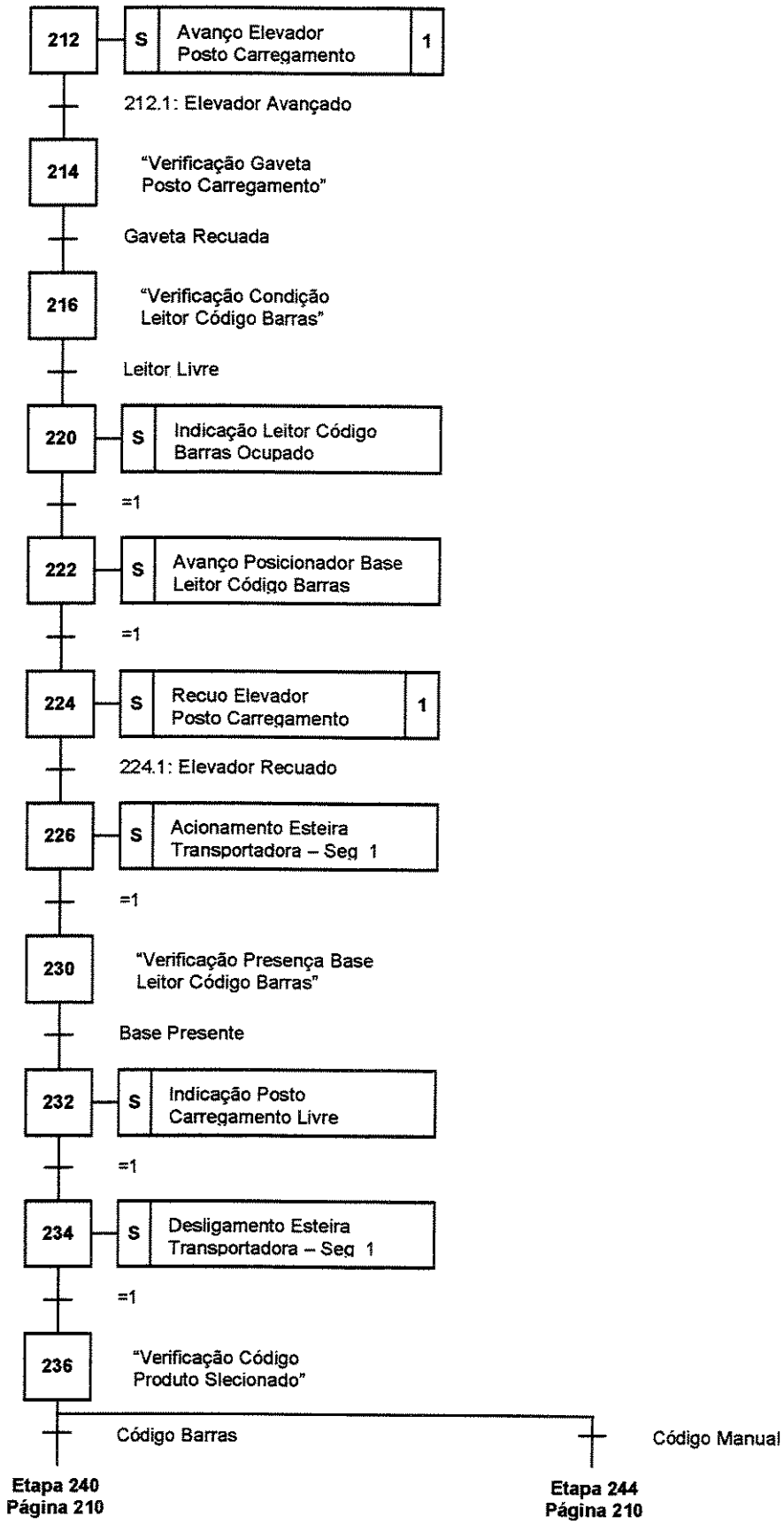

D2-12TR
CA 0 1 2 3 4 5 NC NC NC CB 0 1 2 3 4 5 NC NC
25 MOT EST SEG 001
28 MOT EST SEG 002
32 MOT EST SEG 003
36 MOT EST SEG 004
FONTE EXTERNA 24 VCC
C1
EMERGÊNCIA
FONTE EXTERNA 24 VCC
C1
Title PIPEFA – Sistema de Transferência Módulo Saída – Slot 3 (Y20 – Y37)
Size A
Number
Rev
Date 01 de Novembro de 1999
Drawn by Georgini, M.
Filename PIPEFAST.S06
Sheet 6 of 6

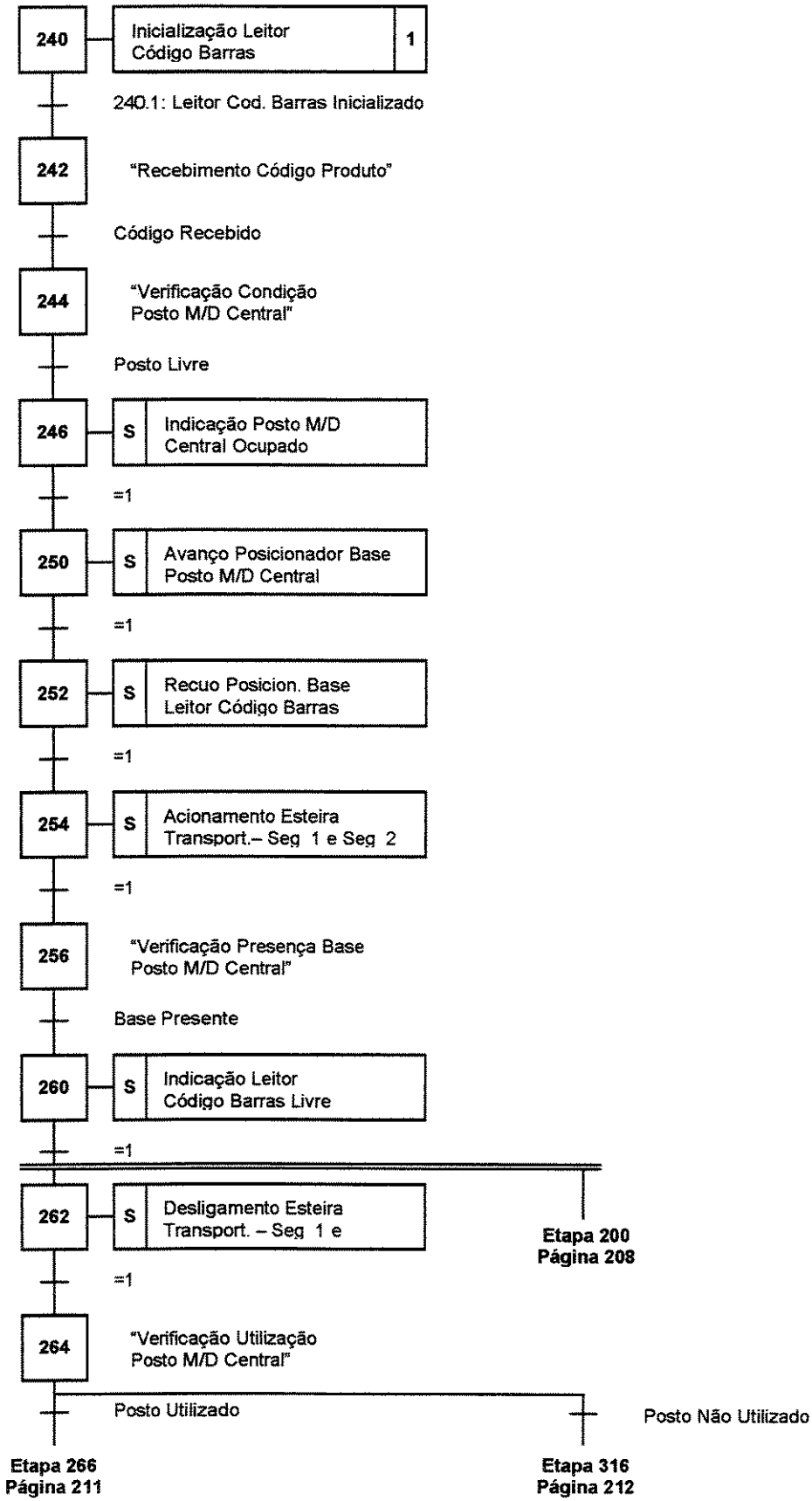
Anexo V

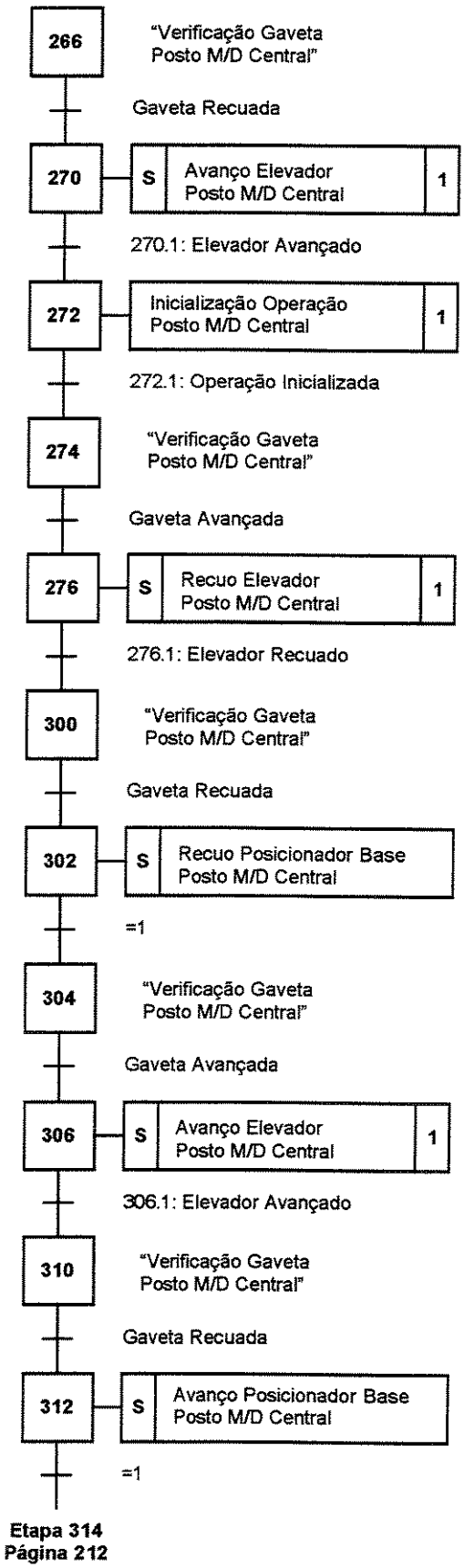
Diagrama Funcional Seqüencial (Sistema de Transferência)

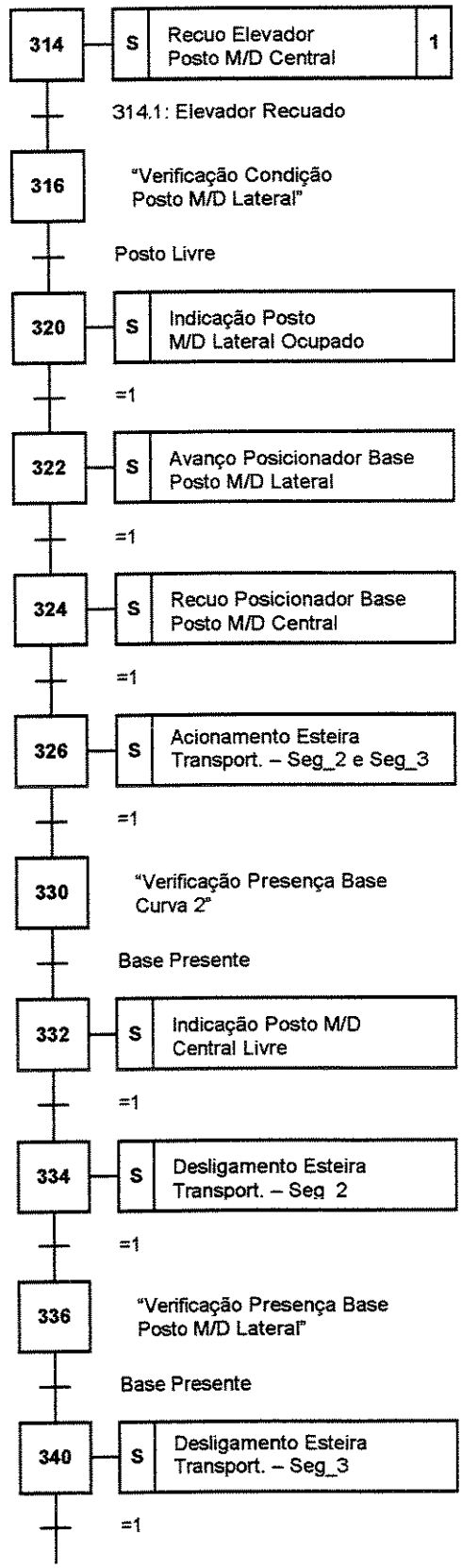
A seguir é apresentado o Diagrama Funcional Seqüencial (*SFC*) relativo à operação do Sistema de Transferência em Modo Automático. As refrências utilizadas em cada Etapa foram mantidas na implementação do Programa de Aplicação, em Linguagem Ladder (Programação por Estágios).

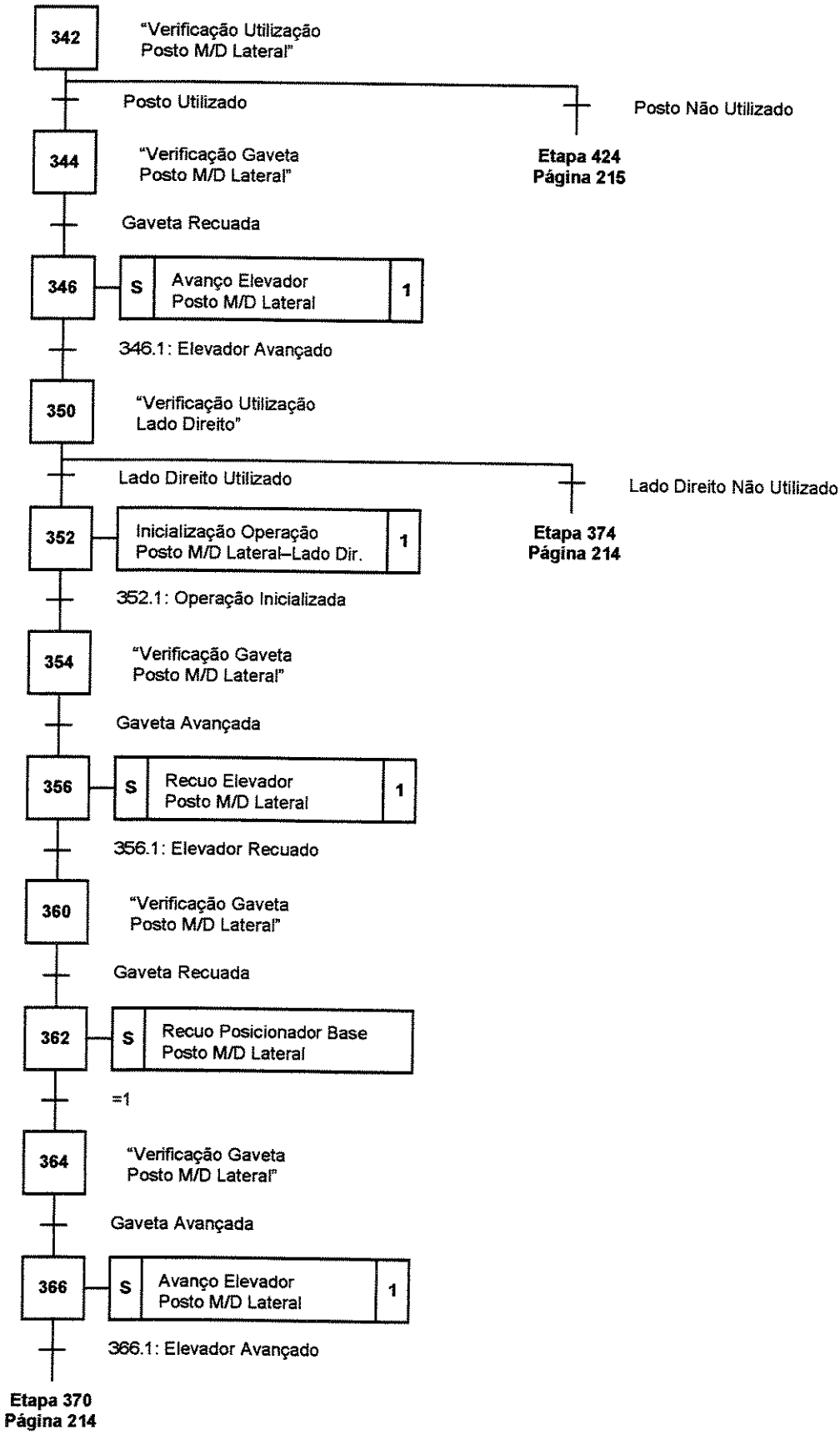


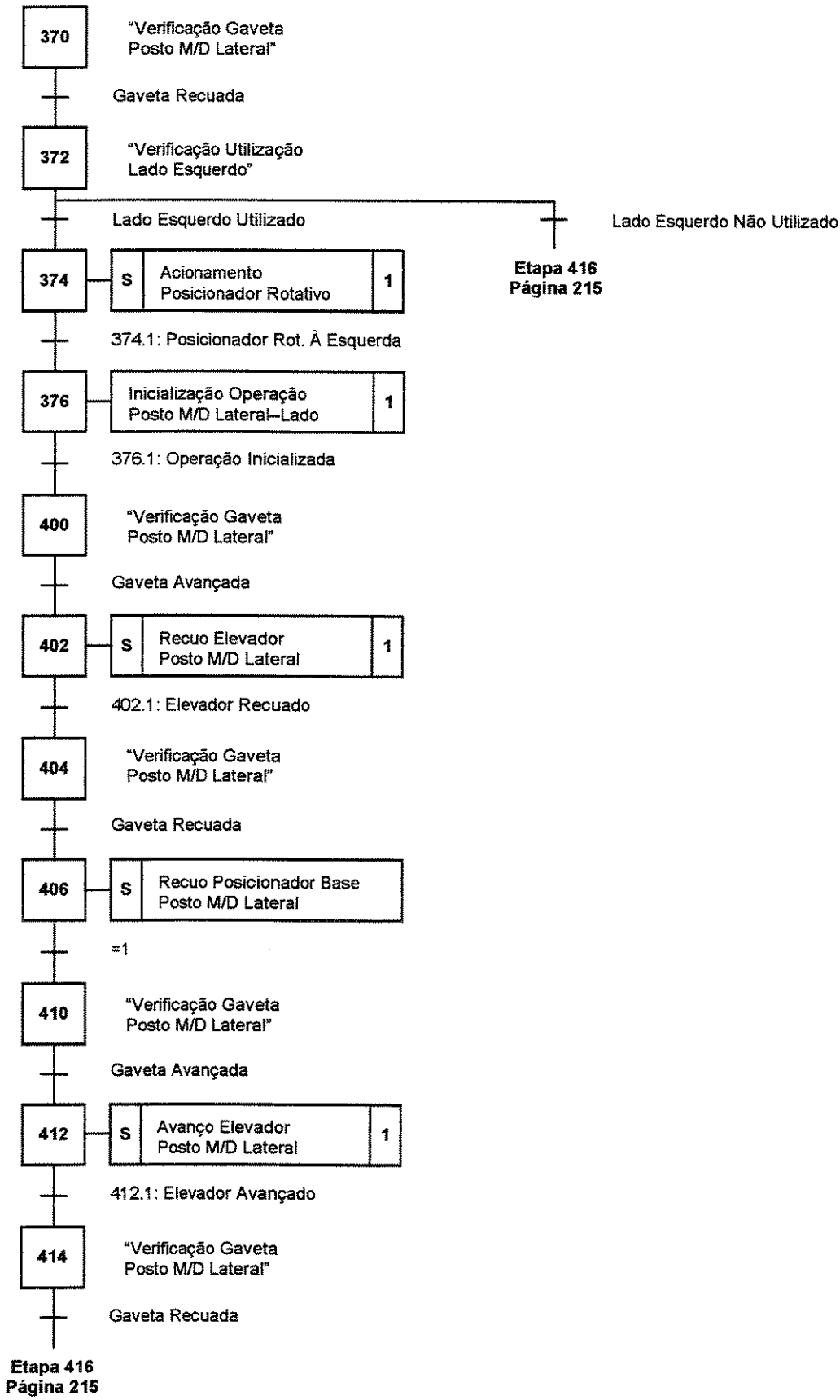

212
S
Avanço Elevador
Posto Carregamento
1
212.1: Elevador Avançado
214
"Verificação Gaveta
Posto Carregamento"
Gaveta Recuada
216
"Verificação Condição
Leitor Código Barras"
Leitor Livre
220
S
Indicação Leitor Código
Barras Ocupado
=1
222
S
Avanço Posicionador Base
Leitor Código Barras
=1
224
S
Recuo Elevador
Posto Carregamento
1
224.1: Elevador Recuado
226
S
Acionamento Esteira
Transportadora – Seg 1
=1
230
"Verificação Presença Base
Leitor Código Barras"
Base Presente
232
S
Indicação Posto
Carregamento Livre
=1
234
S
Desligamento Esteira
Transportadora – Seg 1
=1
236
"Verificação Código
Produto Slecionado"
Código Barras
Código Manual
Etapa 240
Página 210
Etapa 244
Página 210

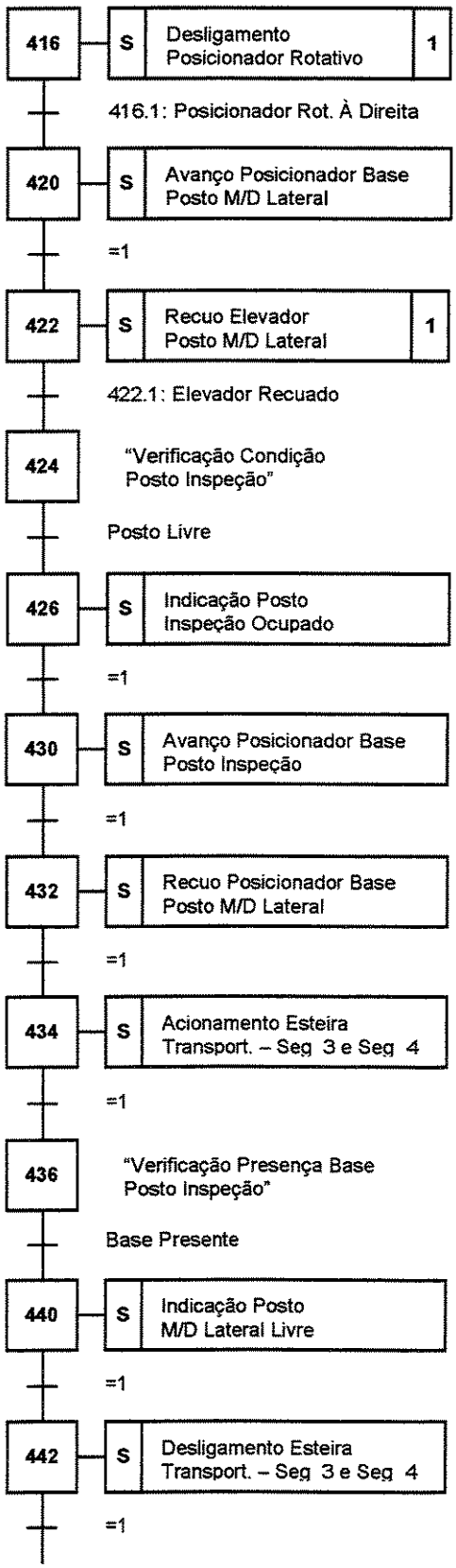

240
Inicialização Leitor Código Barras
1
240.1: Leitor Cod. Barras Inicializado
242
"Recebimento Código Produto"
Código Recebido
244
"Verificação Condição Posto M/D Central"
Posto Livre
246
S
Indicação Posto M/D Central Ocupado
=1
250
S
Avanço Posicionador Base Posto M/D Central
=1
252
S
Recuo Posicion. Base Leitor Código Barras
=1
254
S
Acionamento Esteira Transport.– Seg 1 e Seg 2
=1
256
"Verificação Presença Base Posto M/D Central"
Base Presente
260
S
Indicação Leitor Código Barras Livre
=1
262
S
Desligamento Esteira Transport. – Seg 1 e
=1
264
"Verificação Utilização Posto M/D Central"
Posto Utilizado
Posto Não Utilizado
Etapa 200
Página 208
Etapa 266
Página 211
Etapa 316
Página 212

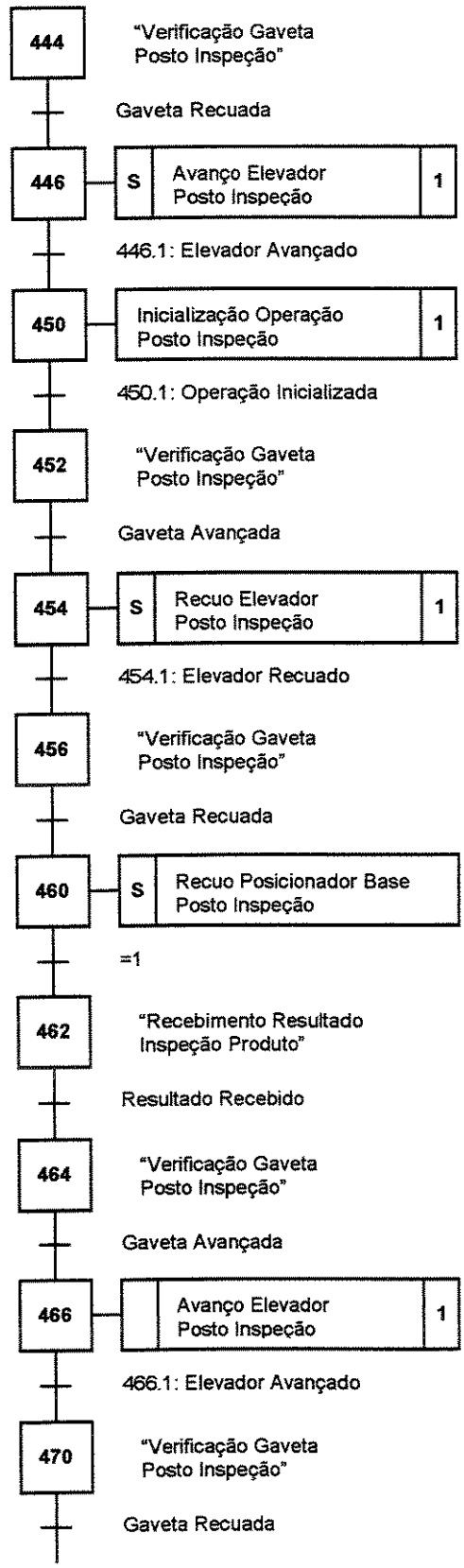

266
"Verificação Gaveta Posto M/D Central"
Gaveta Recuada
270
S
Avanço Elevador Posto M/D Central
1
270.1: Elevador Avançado
272
Inicialização Operação Posto M/D Central
1
272.1: Operação Inicializada
274
"Verificação Gaveta Posto M/D Central"
Gaveta Avançada
276
S
Recuo Elevador Posto M/D Central
1
276.1: Elevador Recuado
300
"Verificação Gaveta Posto M/D Central"
Gaveta Recuada
302
S
Recuo Posicionador Base Posto M/D Central
=1
304
"Verificação Gaveta Posto M/D Central"
Gaveta Avançada
306
S
Avanço Elevador Posto M/D Central
1
306.1: Elevador Avançado
310
"Verificação Gaveta Posto M/D Central"
Gaveta Recuada
312
S
Avanço Posicionador Base Posto M/D Central
=1
Etapa 314
Página 212

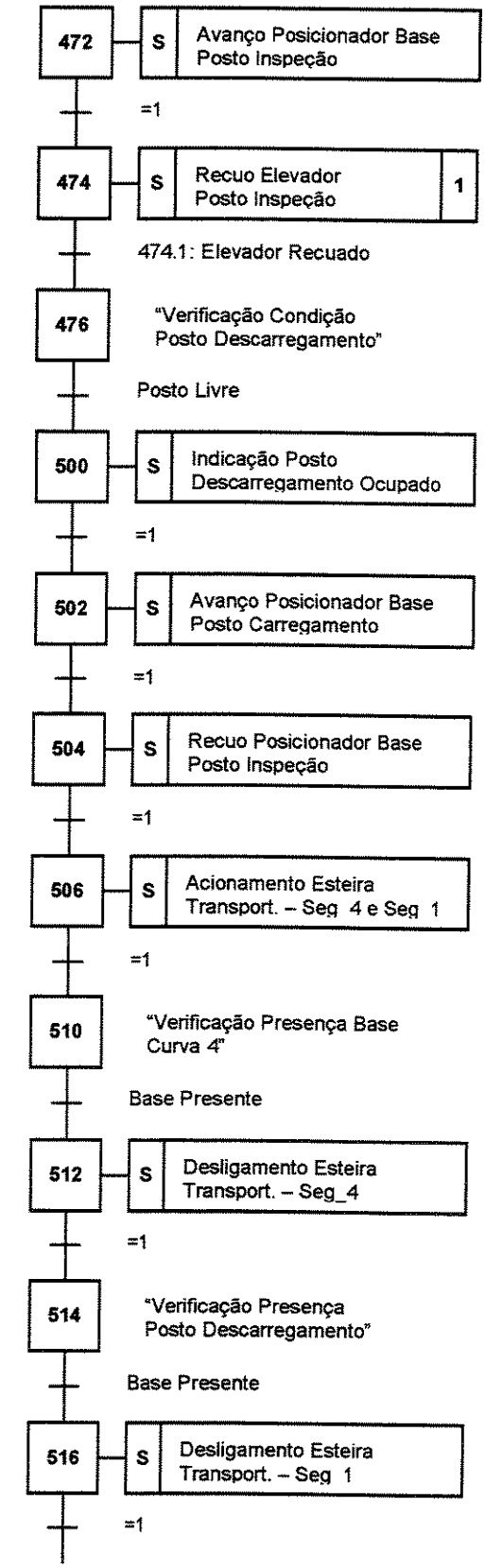

314
S
Recuo Elevador
Posto M/D Central
1
314.1: Elevador Recuado
316
"Verificação Condição
Posto M/D Lateral"
Posto Livre
320
S
Indicação Posto
M/D Lateral Ocupado
=1
322
S
Avanço Posicionador Base
Posto M/D Lateral
=1
324
S
Recuo Posicionador Base
Posto M/D Central
=1
326
S
Acionamento Esteira
Transport. – Seg_2 e Seg_3
=1
330
"Verificação Presença Base
Curva 2"
Base Presente
332
S
Indicação Posto M/D
Central Livre
=1
334
S
Desligamento Esteira
Transport. – Seg_2
=1
336
"Verificação Presença Base
Posto M/D Lateral"
Base Presente
340
S
Desligamento Esteira
Transport. – Seg_3
=1
Etapa 342
Página 213

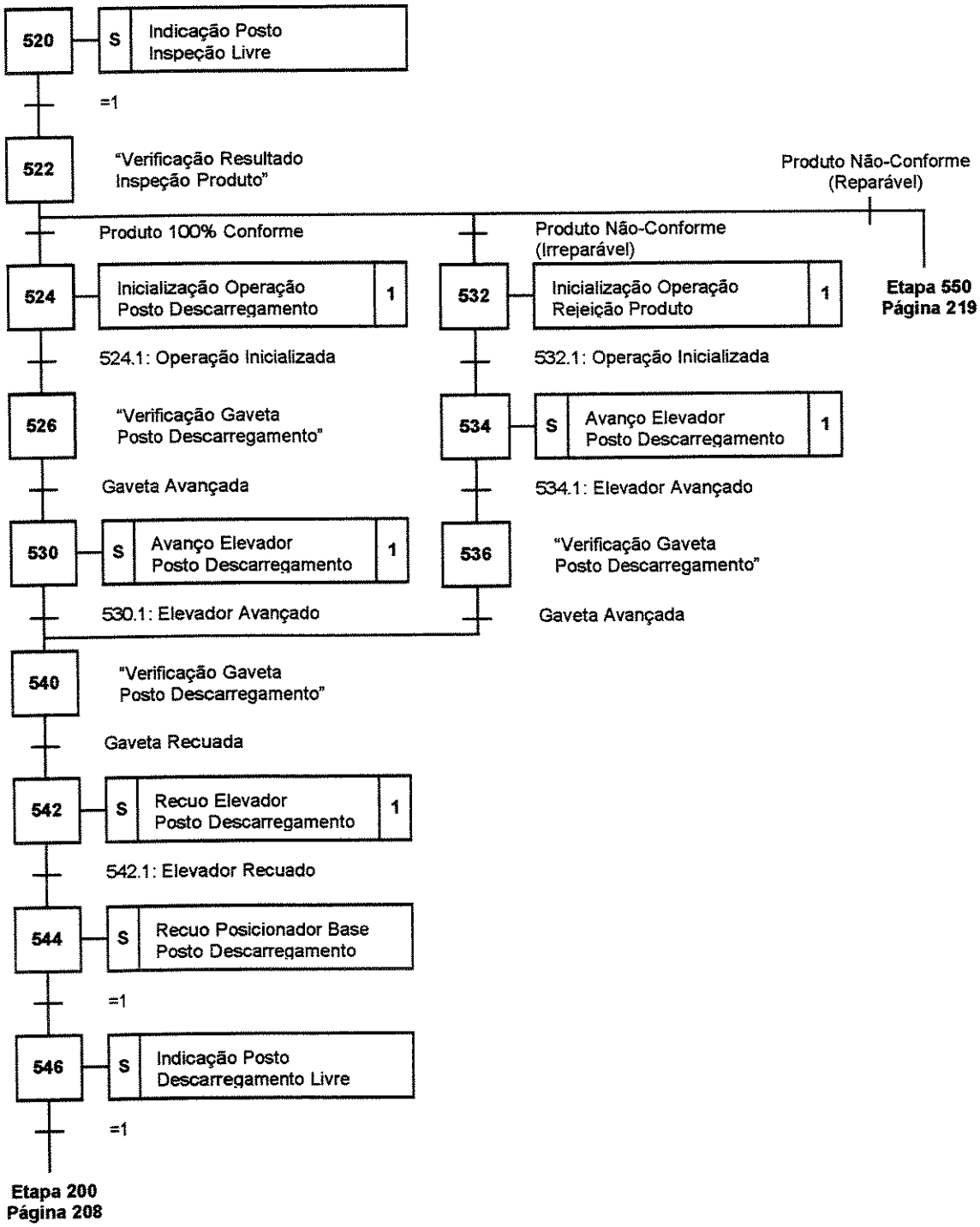

342
"Verificação Utilização Posto M/D Lateral"
Posto Utilizado
Posto Não Utilizado
Etapa 424
Página 215
344
"Verificação Gaveta Posto M/D Lateral"
Gaveta Recuada
346
S
Avanço Elevador Posto M/D Lateral
1
346.1: Elevador Avançado
350
"Verificação Utilização Lado Direito"
Lado Direito Utilizado
Lado Direito Não Utilizado
Etapa 374
Página 214
352
Inicialização Operação Posto M/D Lateral–Lado Dir.
1
352.1: Operação Inicializada
354
"Verificação Gaveta Posto M/D Lateral"
Gaveta Avançada
356
S
Recuo Elevador Posto M/D Lateral
1
356.1: Elevador Recuado
360
"Verificação Gaveta Posto M/D Lateral"
Gaveta Recuada
362
S
Recuo Posicionador Base Posto M/D Lateral
=1
364
"Verificação Gaveta Posto M/D Lateral"
Gaveta Avançada
366
S
Avanço Elevador Posto M/D Lateral
1
366.1: Elevador Avançado
Etapa 370
Página 214

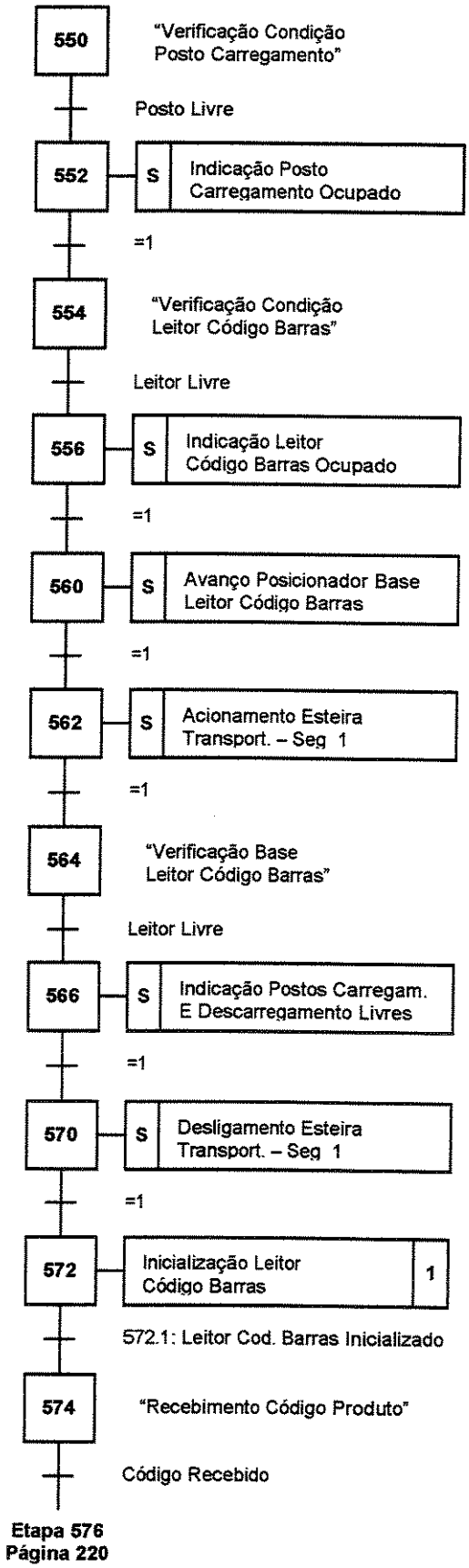

370
"Verificação Gaveta Posto M/D Lateral"
Gaveta Recuada
372
"Verificação Utilização Lado Esquerdo"
Lado Esquerdo Utilizado
Lado Esquerdo Não Utilizado
Etapa 416 Página 215
374
S
Acionamento Posicionador Rotativo
1
374.1: Posicionador Rot. À Esquerda
376
Inicialização Operação Posto M/D Lateral–Lado
1
376.1: Operação Inicializada
400
"Verificação Gaveta Posto M/D Lateral"
Gaveta Avançada
402
S
Recuo Elevador Posto M/D Lateral
1
402.1: Elevador Recuado
404
"Verificação Gaveta Posto M/D Lateral"
Gaveta Recuada
406
S
Recuo Posicionador Base Posto M/D Lateral
=1
410
"Verificação Gaveta Posto M/D Lateral"
Gaveta Avançada
412
S
Avanço Elevador Posto M/D Lateral
1
412.1: Elevador Avançado
414
"Verificação Gaveta Posto M/D Lateral"
Gaveta Recuada
Etapa 416 Página 215

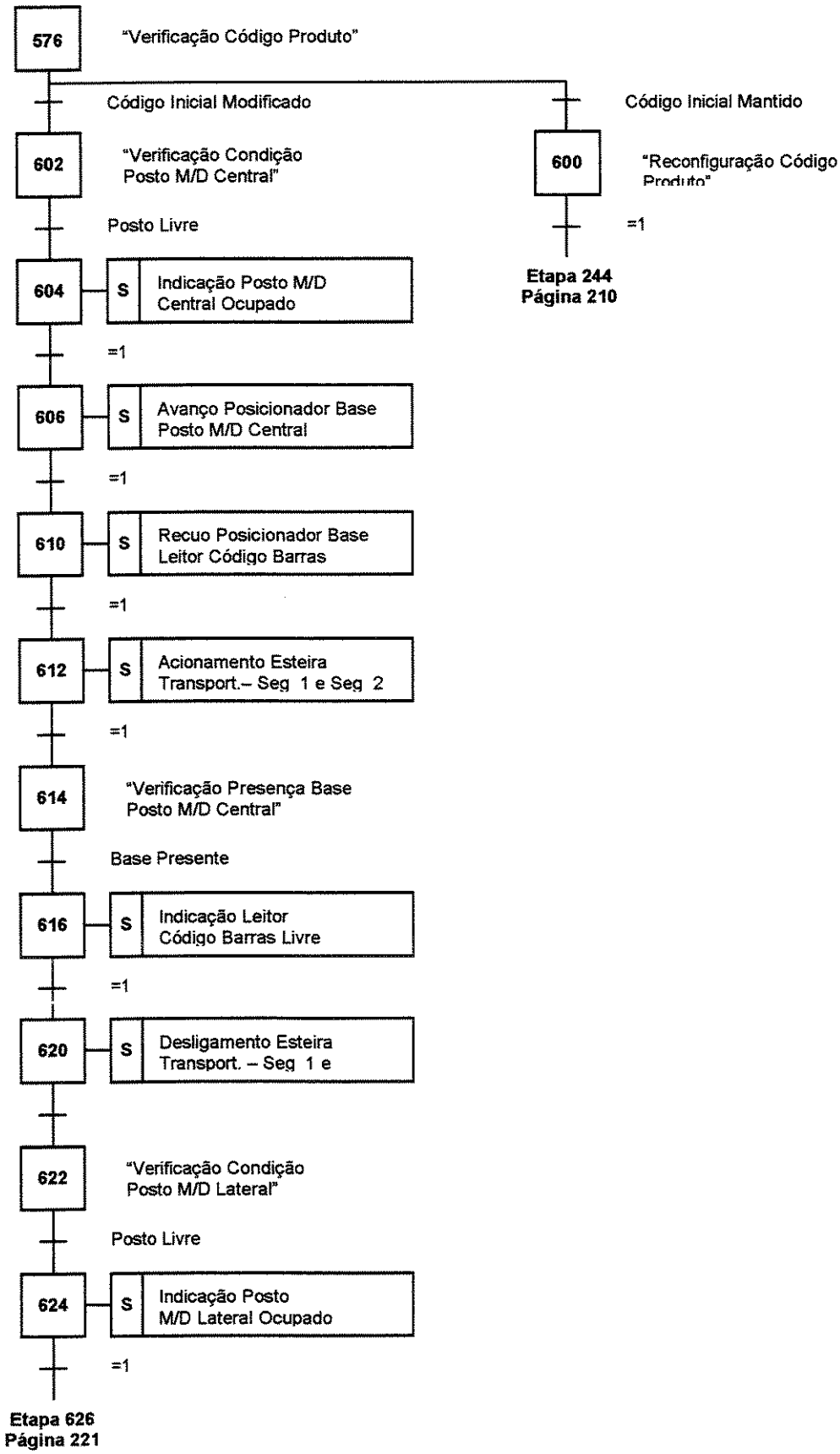

416
S
Desligamento Posicionador Rotativo
1
416.1: Posicionador Rot. À Direita
420
S
Avanço Posicionador Base Posto M/D Lateral
=1
422
S
Recuo Elevador Posto M/D Lateral
1
422.1: Elevador Recuado
424
"Verificação Condição Posto Inspeção"
Posto Livre
426
S
Indicação Posto Inspeção Ocupado
=1
430
S
Avanço Posicionador Base Posto Inspeção
=1
432
S
Recuo Posicionador Base Posto M/D Lateral
=1
434
S
Acionamento Esteira Transport. – Seg 3 e Seg 4
=1
436
"Verificação Presença Base Posto Inspeção"
Base Presente
440
S
Indicação Posto M/D Lateral Livre
=1
442
S
Desligamento Esteira Transport. – Seg 3 e Seg 4
=1
Etapa 444
Página 216

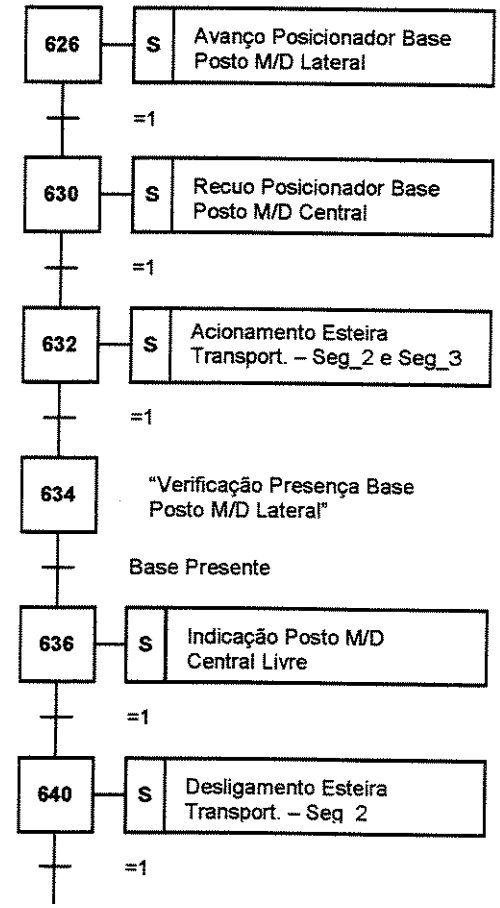

444
"Verificação Gaveta Posto Inspeção"
Gaveta Recuada
446
S
Avanço Elevador Posto Inspeção
1
446.1: Elevador Avançado
450
Inicialização Operação Posto Inspeção
1
450.1: Operação Inicializada
452
"Verificação Gaveta Posto Inspeção"
Gaveta Avançada
454
S
Recuo Elevador Posto Inspeção
1
454.1: Elevador Recuado
456
"Verificação Gaveta Posto Inspeção"
Gaveta Recuada
460
S
Recuo Posicionador Base Posto Inspeção
=1
462
"Recebimento Resultado Inspeção Produto"
Resultado Recebido
464
"Verificação Gaveta Posto Inspeção"
Gaveta Avançada
466
Avanço Elevador Posto Inspeção
1
466.1: Elevador Avançado
470
"Verificação Gaveta Posto Inspeção"
Gaveta Recuada
Etapa 472
Página 217


472
S
Avanço Posicionador Base
Posto Inspeção
=1
474
S
Recuo Elevador
Posto Inspeção
1
474.1: Elevador Recuado
476
"Verificação Condição
Posto Descarregamento"
Posto Livre
500
S
Indicação Posto
Descarregamento Ocupado
=1
502
S
Avanço Posicionador Base
Posto Carregamento
=1
504
S
Recuo Posicionador Base
Posto Inspeção
=1
506
S
Acionamento Esteira
Transport. – Seg 4 e Seg 1
=1
510
"Verificação Presença Base
Curva 4"
Base Presente
512
S
Desligamento Esteira
Transport. – Seg_4
=1
514
"Verificação Presença
Posto Descarregamento"
Base Presente
516
S
Desligamento Esteira
Transport. – Seg 1
=1
Etapa 520
Página 218


520
S
Indicação Posto
Inspeção Livre
=1
522
"Verificação Resultado
Inspeção Produto"
Produto Não-Conforme
(Reparável)
Produto 100% Conforme
Produto Não-Conforme
(Irreparável)
524
Inicialização Operação
Posto Descarregamento
1
532
Inicialização Operação
Rejeição Produto
1
Etapa 550
Página 219
524.1: Operação Inicializada
532.1: Operação Inicializada
526
"Verificação Gaveta
Posto Descarregamento"
534
S
Avanço Elevador
Posto Descarregamento
1
Gaveta Avançada
534.1: Elevador Avançado
530
S
Avanço Elevador
Posto Descarregamento
1
536
"Verificação Gaveta
Posto Descarregamento"
530.1: Elevador Avançado
Gaveta Avançada
540
"Verificação Gaveta
Posto Descarregamento"
Gaveta Recuada
542
S
Recuo Elevador
Posto Descarregamento
1
542.1: Elevador Recuado
544
S
Recuo Posicionador Base
Posto Descarregamento
=1
546
S
Indicação Posto
Descarregamento Livre
=1
Etapa 200
Página 208


550
"Verificação Condição
Posto Carregamento"
Posto Livre
552
S
Indicação Posto
Carregamento Ocupado
=1
554
"Verificação Condição
Leitor Código Barras"
Leitor Livre
556
S
Indicação Leitor
Código Barras Ocupado
=1
560
S
Avanço Posicionador Base
Leitor Código Barras
=1
562
S
Acionamento Esteira
Transport. – Seg 1
=1
564
"Verificação Base
Leitor Código Barras"
Leitor Livre
566
S
Indicação Postos Carregam.
E Descarregamento Livres
=1
570
S
Desligamento Esteira
Transport. – Seg 1
=1
572
Inicialização Leitor
Código Barras
1
572.1: Leitor Cod. Barras Inicializado
574
"Recebimento Código Produto"
Código Recebido
Etapa 576
Página 220


576
"Verificação Código Produto"
Código Inicial Modificado
Código Inicial Mantido
602
"Verificação Condição Posto M/D Central"
600
"Reconfiguração Código Produto"
Posto Livre
=1
604
S
Indicação Posto M/D Central Ocupado
Etapa 244 Página 210
=1
606
S
Avanço Posicionador Base Posto M/D Central
=1
610
S
Recuo Posicionador Base Leitor Código Barras
=1
612
S
Acionamento Esteira Transport.– Seg 1 e Seg 2
=1
614
"Verificação Presença Base Posto M/D Central"
Base Presente
616
S
Indicação Leitor Código Barras Livre
=1
620
S
Desligamento Esteira Transport. – Seg 1 e
622
"Verificação Condição Posto M/D Lateral"
Posto Livre
624
S
Indicação Posto M/D Lateral Ocupado
=1
Etapa 626 Página 221



Etapa 424
Página 215

Anexo VI

Execução do Software *Interact* (Aplicação PIPEFA)

Para execução do Software *Interact* é necessário a utilização de *Hard Key* apropriada. No entanto, há a opção de demonstração (*Preview Mode*) que permite a utilização de todos os recursos do Software (Desenvolvimento e *Runtime*) pelo período de uma hora.

Ao executar o Software *Interact*, sem que a *Hard Key* esteja instalada, é apresentada a janela figura A6.1, bastando apenas selecionar a opção *Preview* e concordar (*I Agree*) com as condições apresentadas em seguida (utilização do software apenas para efeito de 'teste').

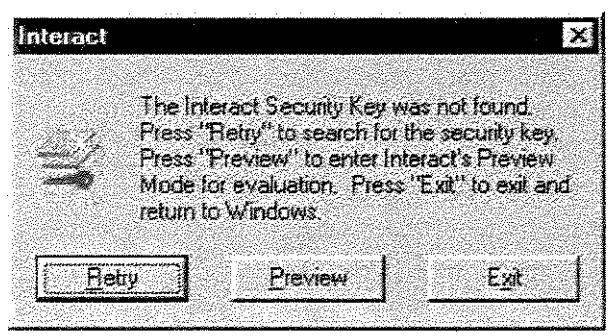


Figura A6.1 – Janela Inicial *Interact* (*Preview Mode*)

Porém, para que seja possível executar a Aplicação desenvolvida (*Runtime*), faz-se necessário que a opção '*Time Zone*' do relógio do Windows® esteja configurado para *(GMT* –

05:00) Eastern Time (US & Canada) e que a opção de ajuste automático para horário de verão esteja desabilitado, conforme apresentado na figura A6.2.

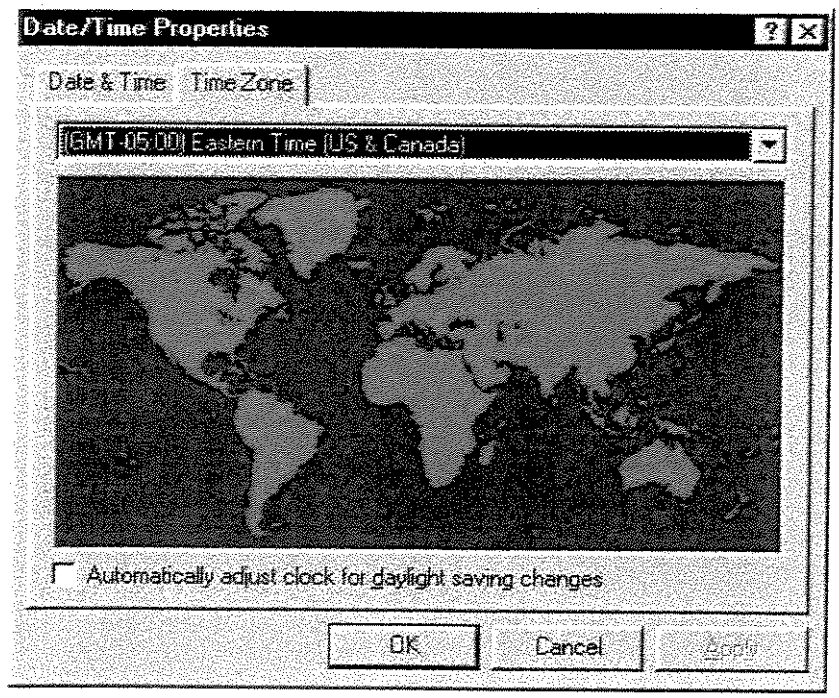


Figura A6.2 – Ajuste Calendário Relógio (Windows®)

Outro detalhe importante quanto à execução do Software *Interact*, e que deve ser observado, relaciona-se à configuração da porta serial do PC utilizado. Esta deve estar habilitada tanto no hardware, como no software (*Set Up* do PC e Windows®).

Recomenda-se que o PC utilizado para rodar o Software *Interact* tenha o Windows® 95 na versão em inglês, para evitar possíveis falhas (travamento do sistema, por exemplo) apresentadas ao executá-lo em PCs com Windows® 95 em Português.