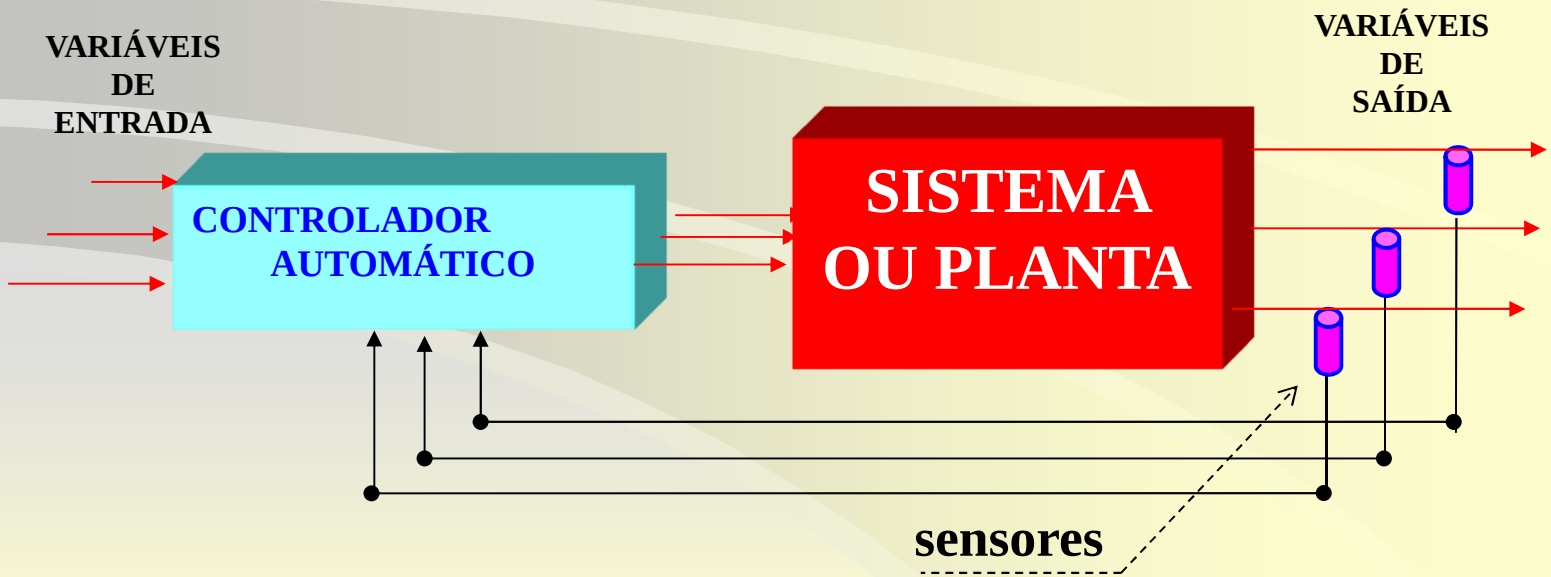
ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PEA - DEPARTAMENTO DE ENGENHARIA DE ENERGIA E AUTOMAÇÃO ELÉTRICAS

APOSTILA
LABORATÓRIO DE AUTOMAÇÃO INDUSTRIAL
TEORIA GERAL DE AUTOMAÇÃO INDUSTRIAL
CONTROLADORES LÓGICOS PROGRAMÁVEIS E CAP/CAP
(Controlador de Automação Programável)
(Programmable Automation Controller)
2016

Prof. Dr. Sergio Luiz Pereira

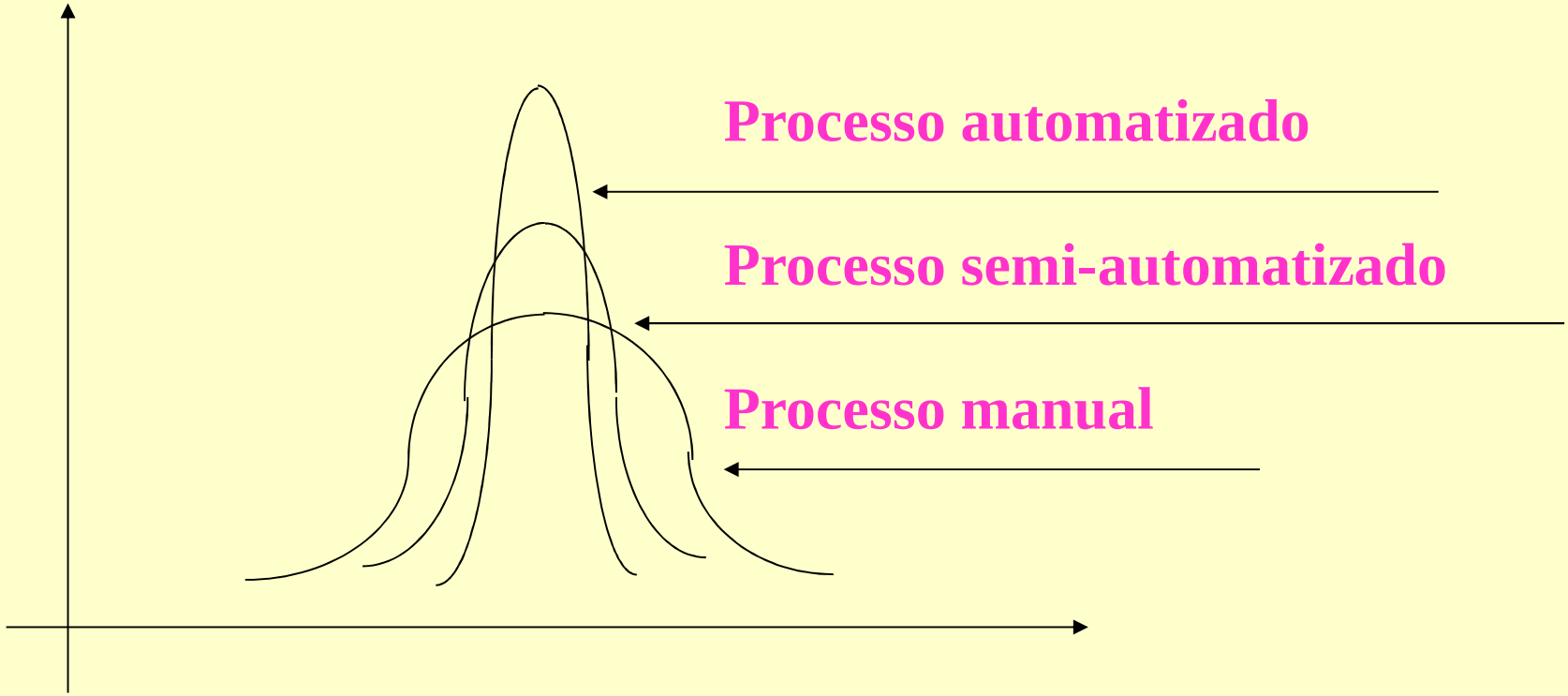

CONJUNTO u(t) DE VARIÁVEIS DE ENTRADA OU CONTROLADORAS
{u1(t),.....,up(t)} t0≤t≤tf
SISTEMA OU PLANTA
CONJUNTO y(t) DE VARIÁVEIS DE SAÍDA OU CONTROLADAS
{y1(t),.....,yp(t)}
t0≤t≤tf


VARIÁVEIS DE ENTRADA
CONTROLADOR AUTOMÁTICO
SISTEMA OU PLANTA
VARIÁVEIS DE SAÍDA
sensores

AS VANTAGENS SOBRE O CONTROLE HUMANO PODEM SER AGRUPADAS EM SETE CLASSIFICAÇÕES:

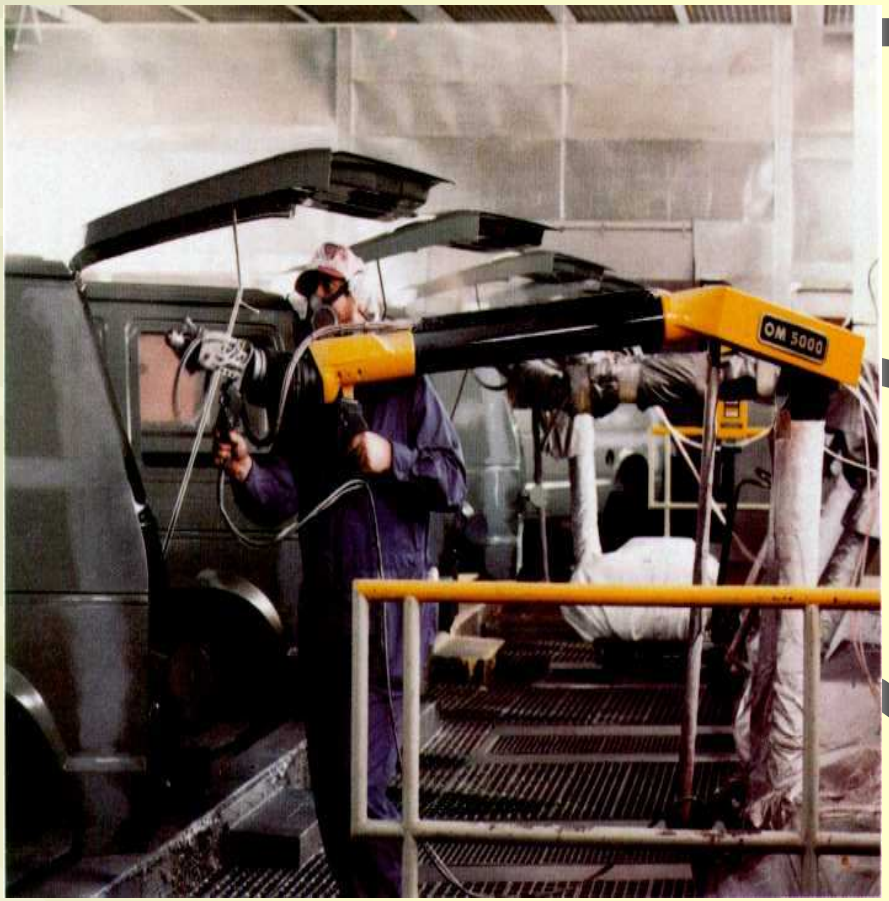
1 - REDUÇÃO DOS CUSTOS DE PRODUÇÃO.
2 - CAPACIDADE COMPUTACIONAL DISPONÍVEL.
3 - RÁPIDA RESPOSTA.
4 - REDUÇÃO DO TAMANHO E CUSTO DO EQUIPAMENTO.
5 - SEGURANÇA AMBIENTAL E SEGURANÇA HUMANA.
6 - RECONHECIMENTO E REAÇÃO IMEDIATA EM SITUAÇÃO EMERGENCIAL.
7 - GARANTIA DA QUALIDADE.

Processos automatizados possuem uma série de vantagens em relação aos processos manuais. Uma delas é a garantia da qualidade.

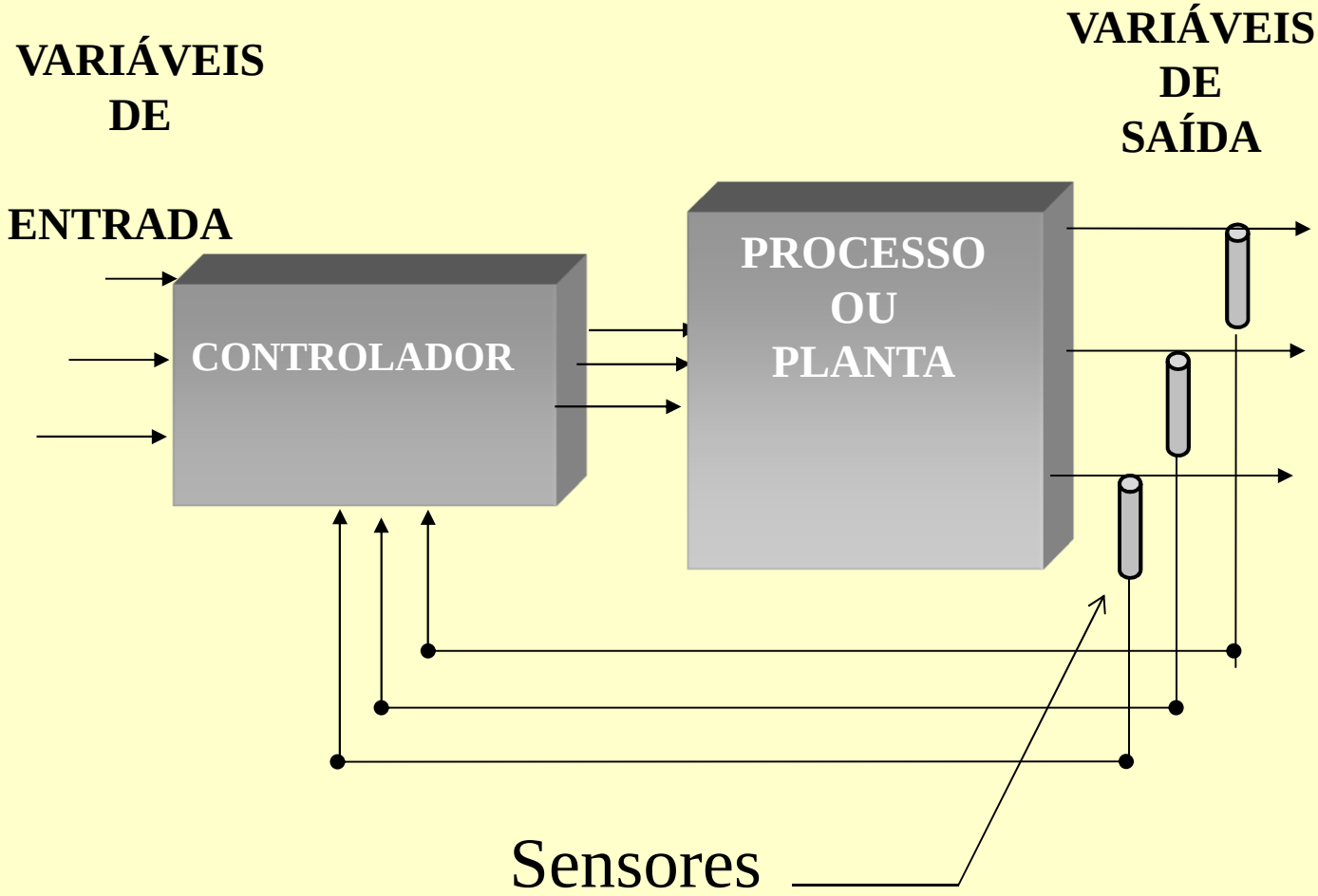


AUTOMAÇÃO DE PROCESSOS

Ex. de Processo Automatizado

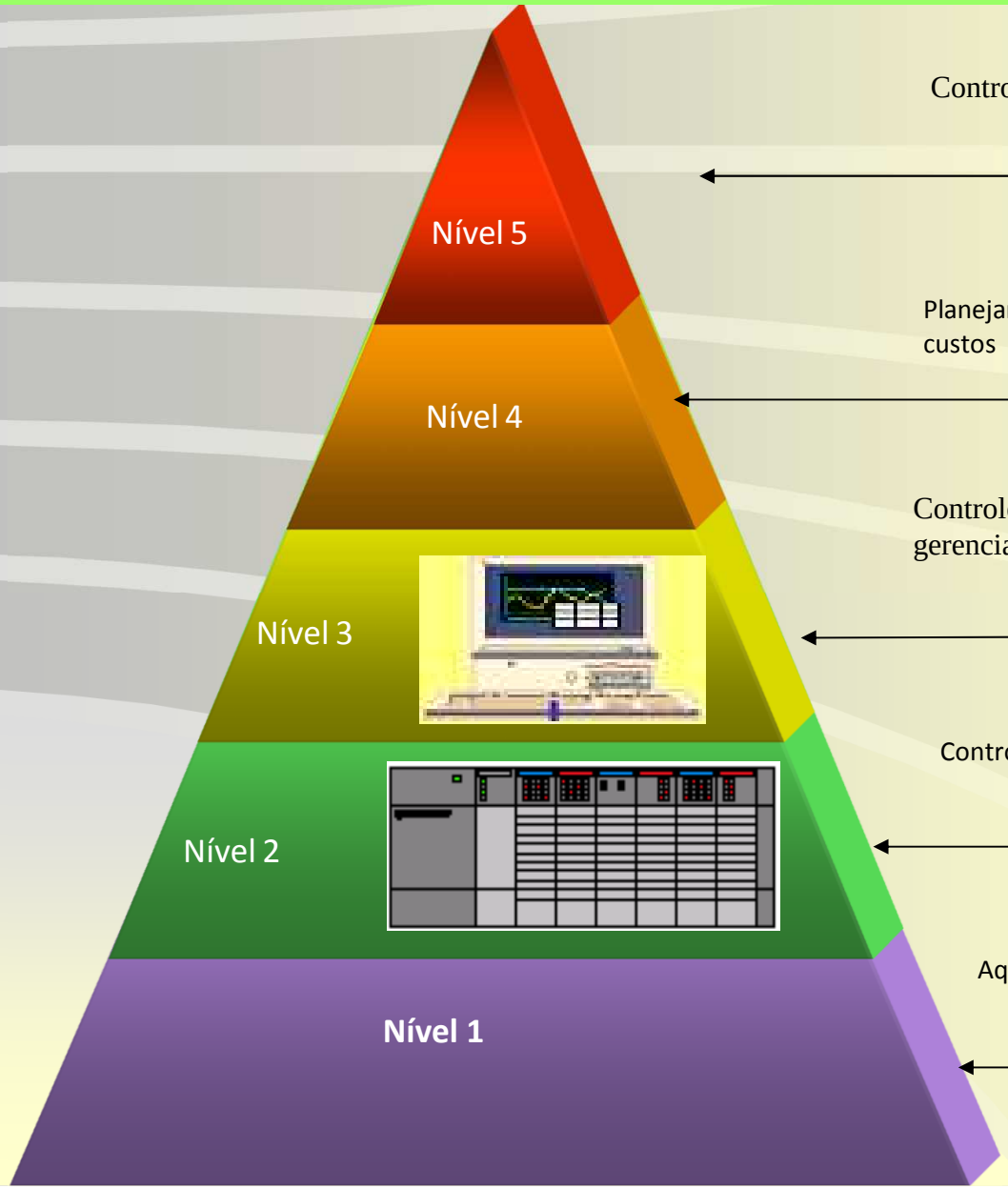


Ex de Processo Manual



Ex. de Processo Automatizado

NÍVEIS DE AUTOMAÇÃO INDUSTRIAL E TIPOS DE PROCESSOS



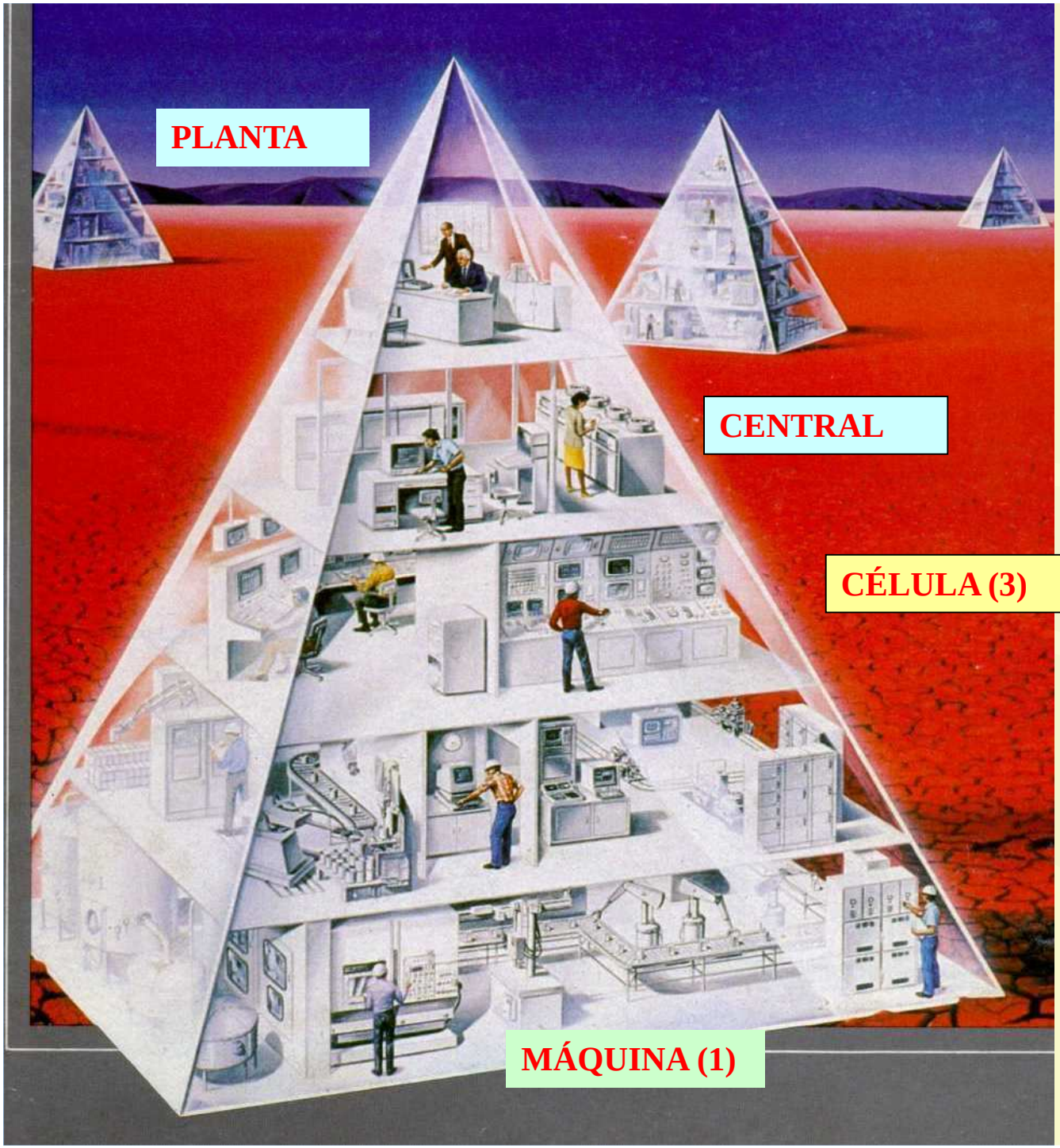
Controle fabril total, produção e programação

Planejamento Estratégico Controle sobre vendas e custos

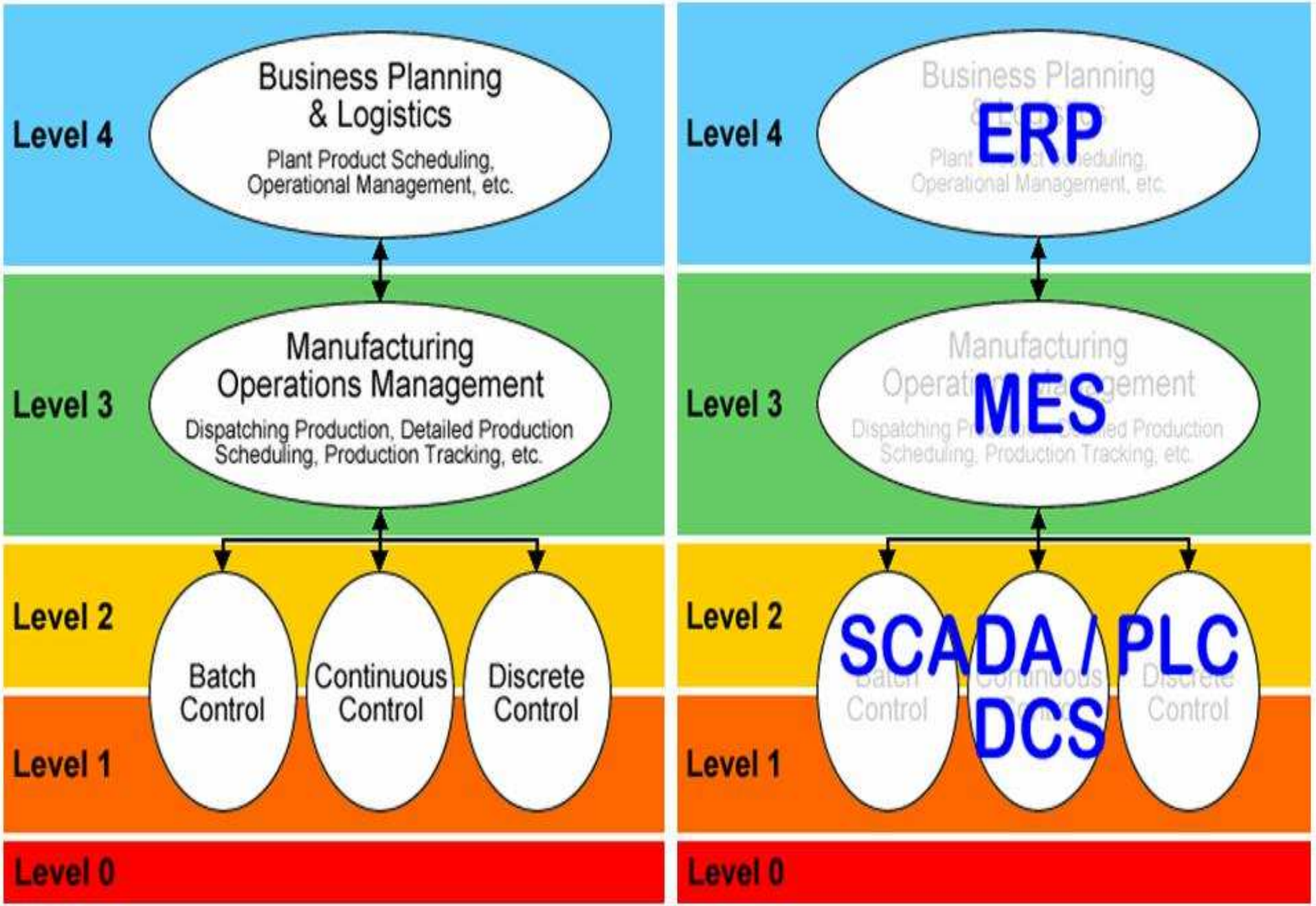
Controle de grupo (célula de trabalho) gerenciamento e Otimização de Processo

Controle individual (PLCs, Inversores, reles etc)

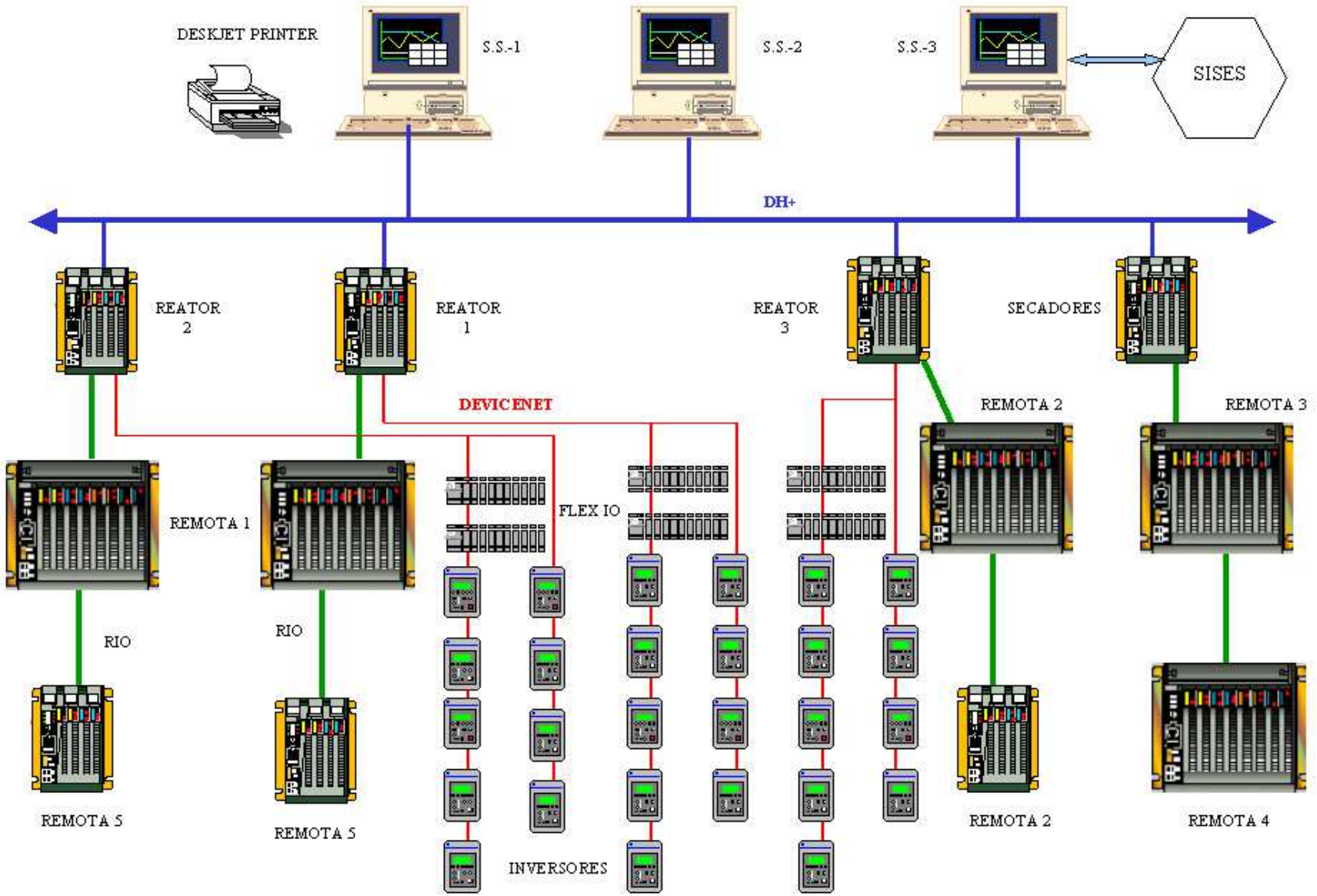
Aquisição de Dados e Controle Manual

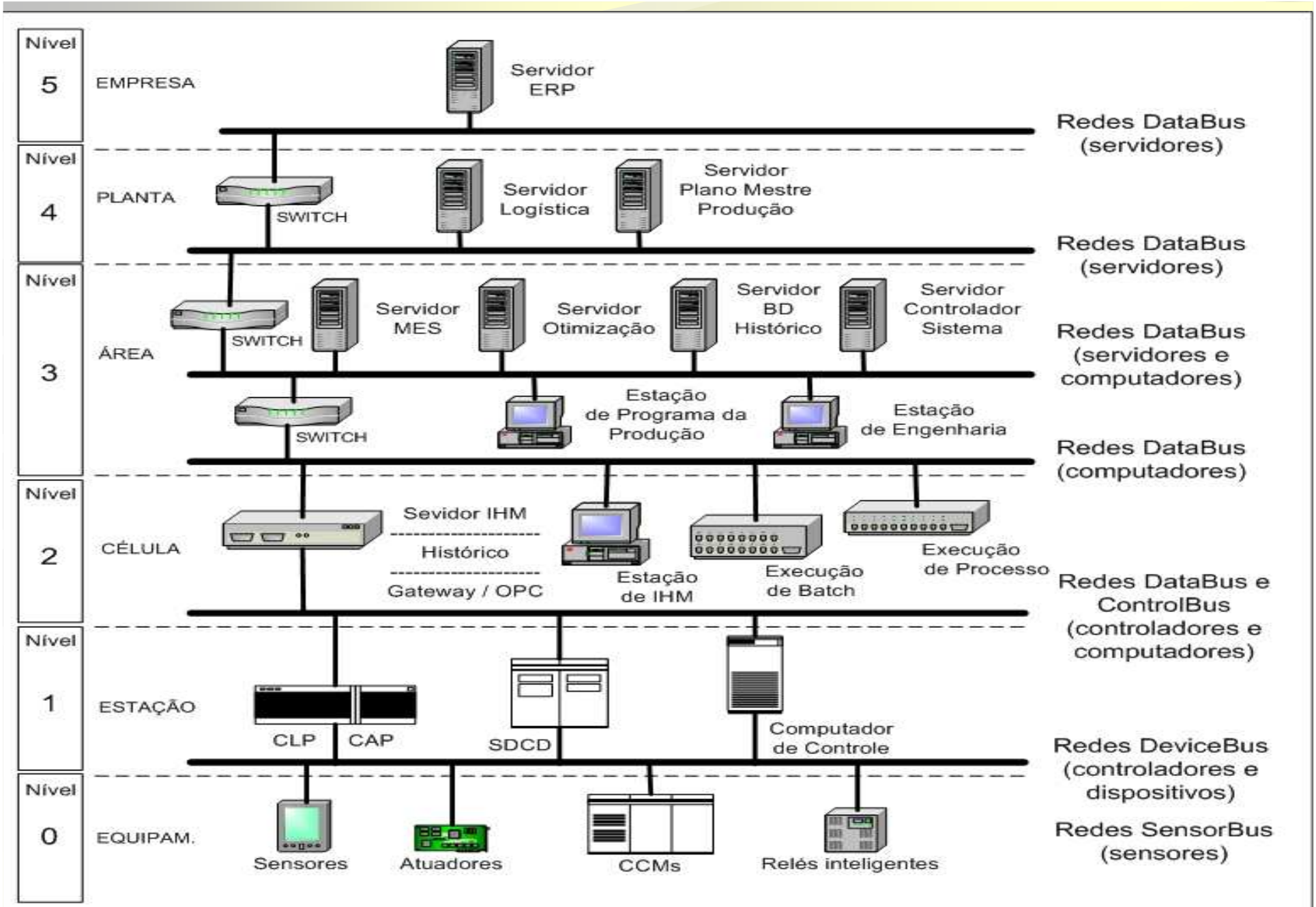


CONCEITO DA PIRÂMIDE



Visão Funcional Fonte: (ISA 95).

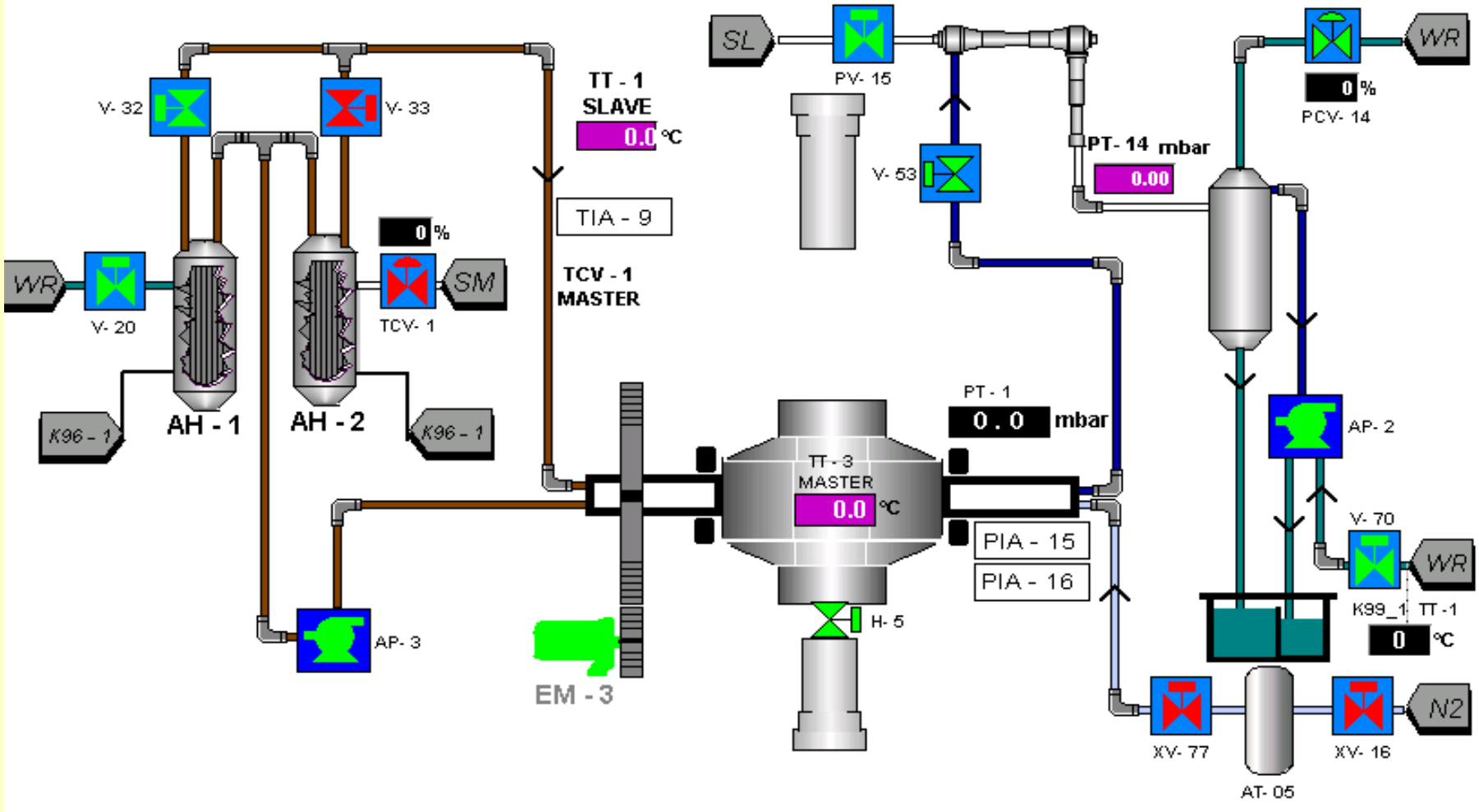

DESKJET PRINTER
S.S.-1
S.S.-2
S.S.-3
SISES
DH+
REATOR 2
REATOR 1
REATOR 3
SECADORES
DEVICENET
REMOTA 2
REMOTA 3
REMOTA 1
FLEX IO
RIO
RIO
REMOTA 5
REMOTA 5
INVERSORES
REMOTA 2
REMOTA 4



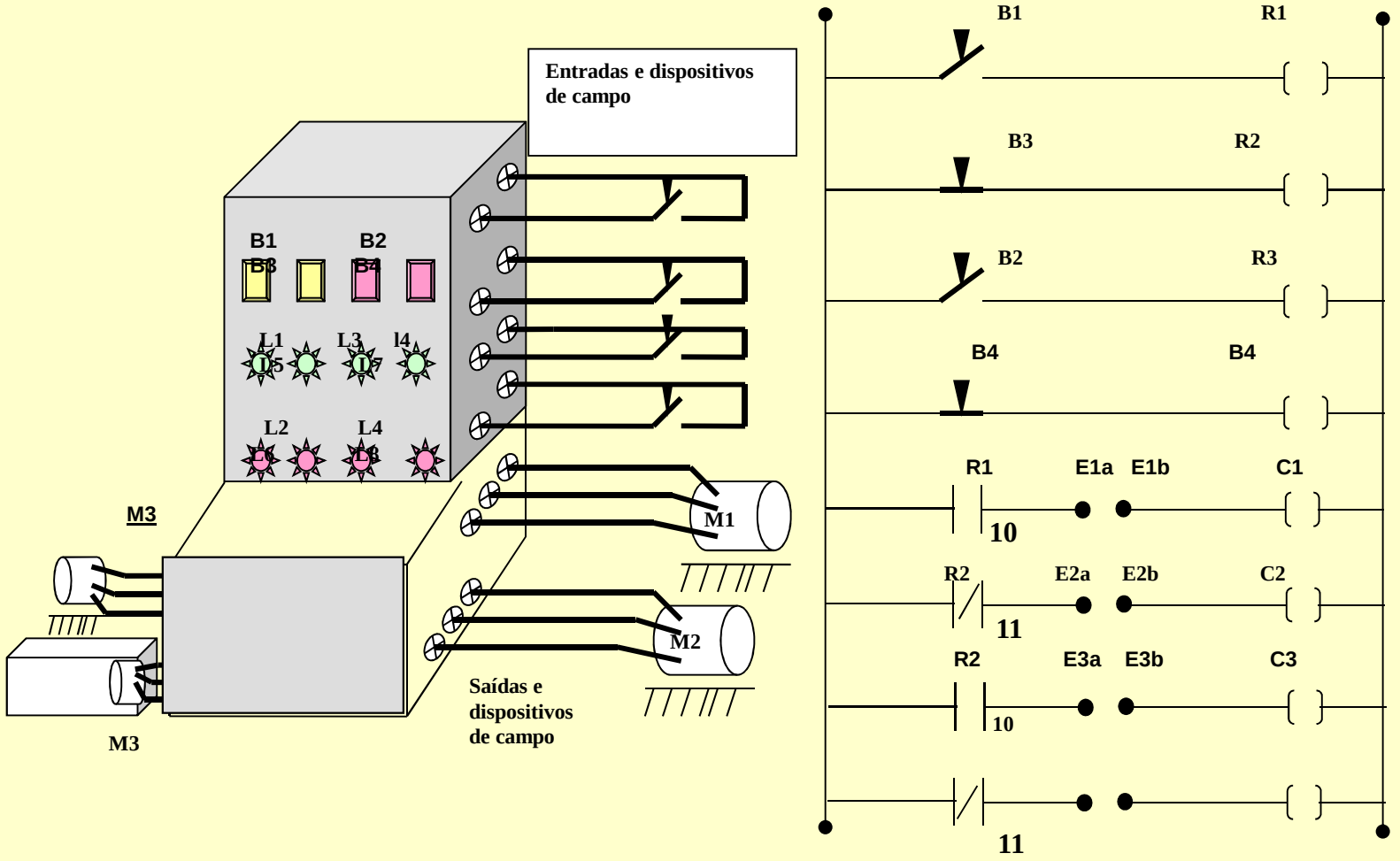
Arquitetura de hardware genérica dentro do modelo da pirâmide da automação.

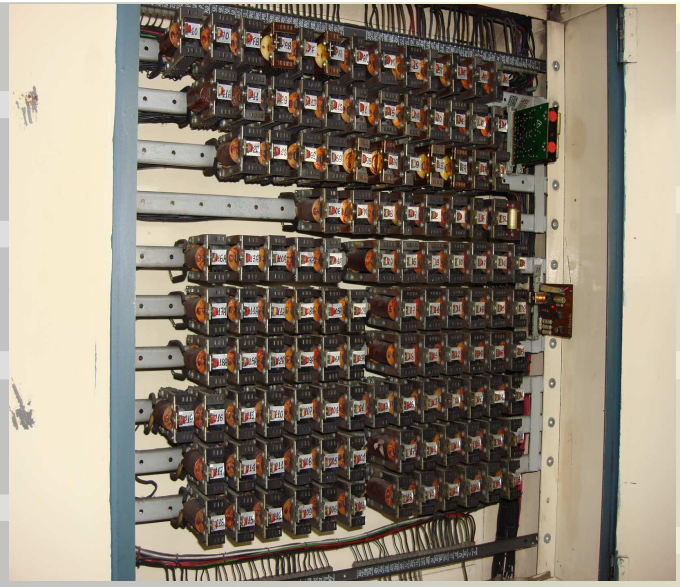
MODULARIZAÇÃO DE PRODUTOS DEPENDEM PRINCIPALMENTE DE PROJETOS E DE PROCESSOS.

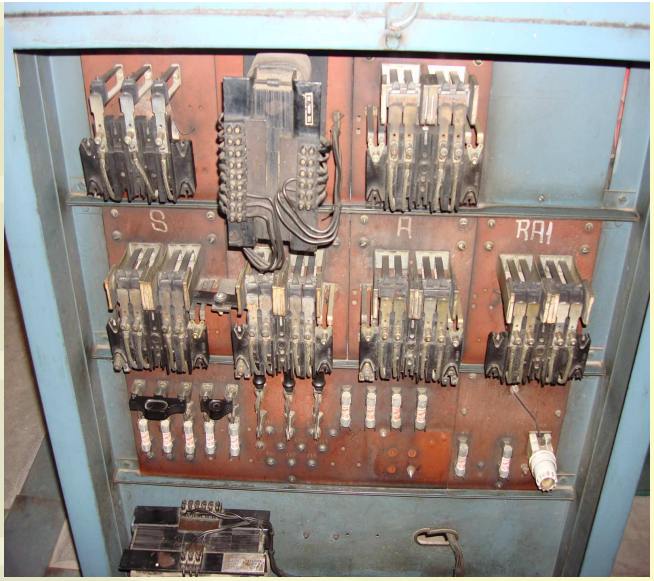
MODULARIZAÇÃO DE PROCESSOS DEPENDEM DE MÉTODOS E DE AUTOMAÇÃO DE PROCESSOS.



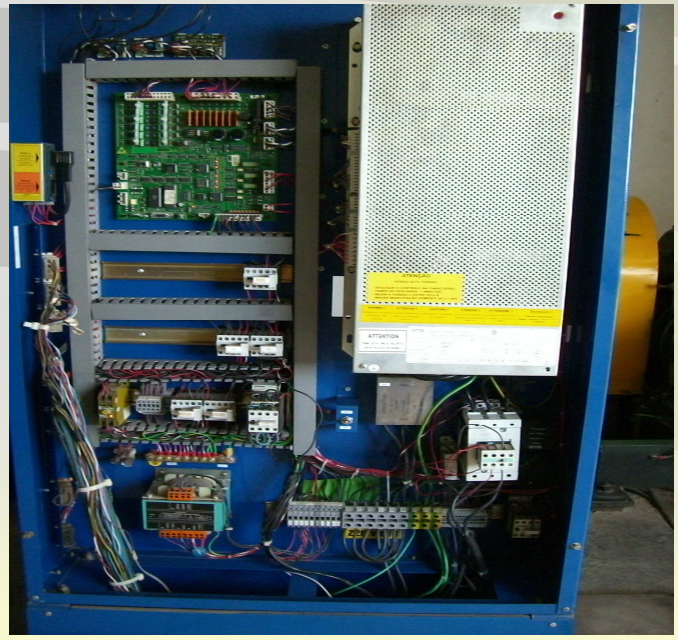
Exemplo de um painel genérico de comando e de parte do seu circuito elétrico. (Automação eletromecânica)

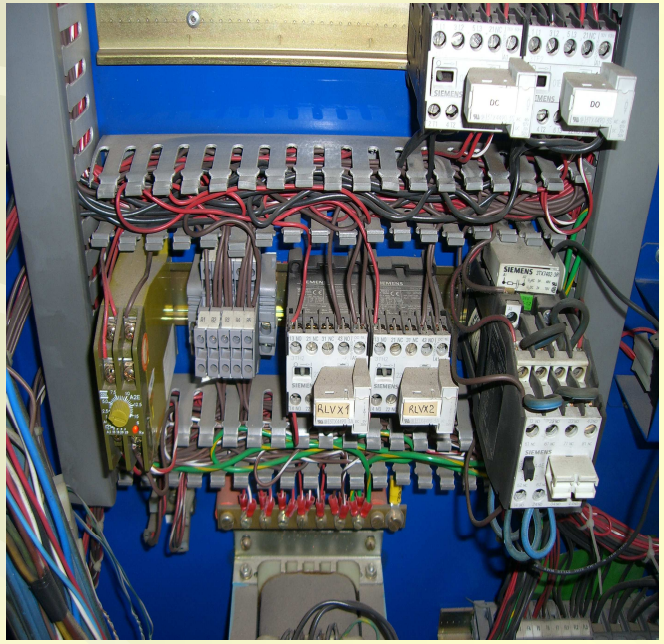






Fotografias de um painel de comando eletromecânico para elevadores.



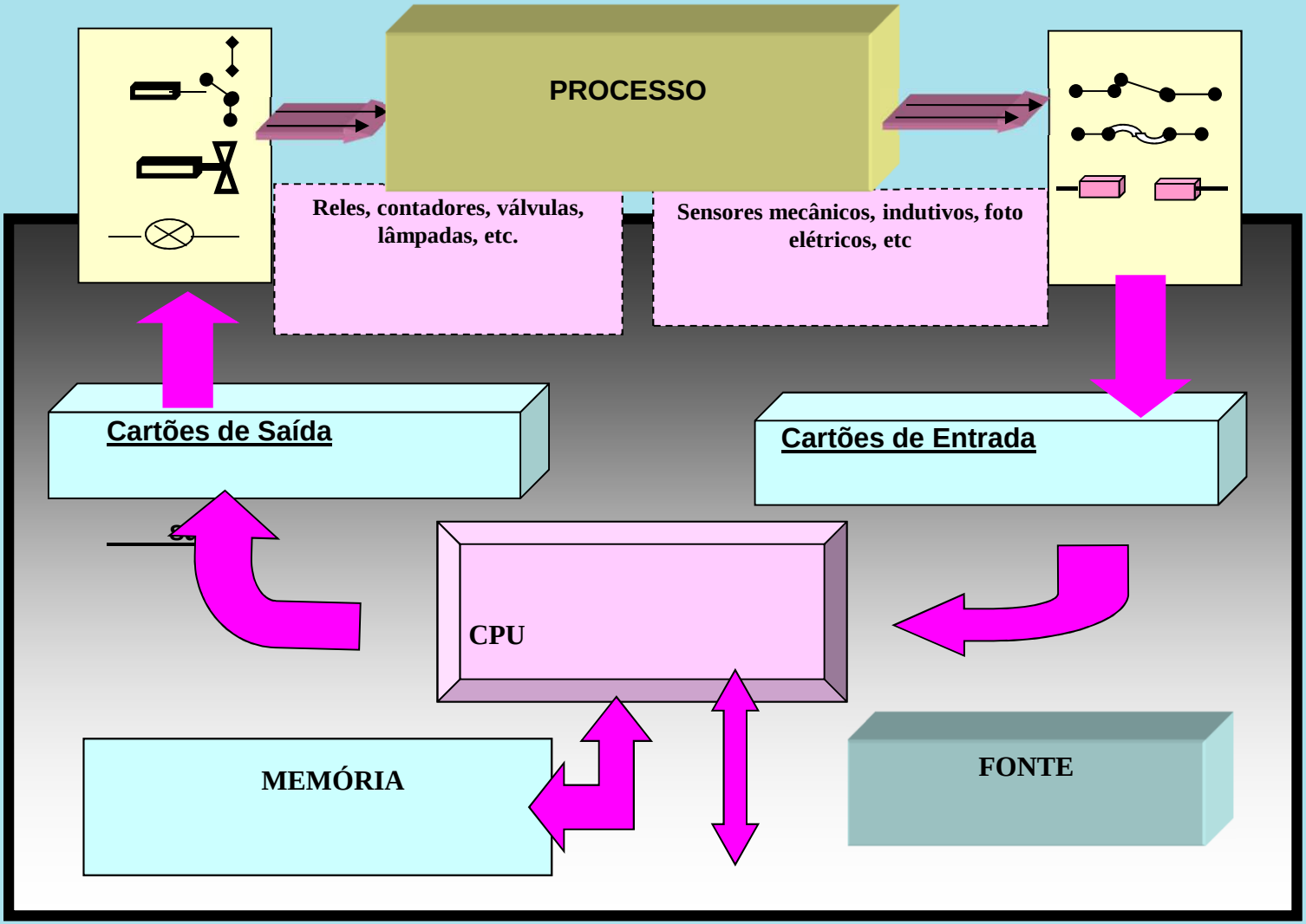


Fotografias de um painel de comando eletromecânico e também micro processado que emprega inversor de freqüência para o acionamento do motor do elevador.

Histórico dos Controladores Lógicos Programáveis

O desenvolvimento do CLPs (Controladores Lógicos Programáveis) ou CPs (Controladores Programáveis) começou por volta de 1968 quando a General Motors solicitou à indústria eletrônica uma alternativa para a lógica eletromecânica baseada em relés. Os sistemas de relés utilizados na manufatura, montagem, carregamento e controle de máquinas haviam se tornado muito grandes e complexos, aumentando os custos de manutenção e baixando a confiabilidade. Outro problema era a grande complexidade envolvida em qualquer mudança na planta industrial ou produtiva.

ARQUITETURA BÁSICA DE HARDWARE DE UM CLP



DISPOSITIVO DE PROGRAMAÇÃO - IMPRESSORA -MONITOR - DISPLAY

Fonte:

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

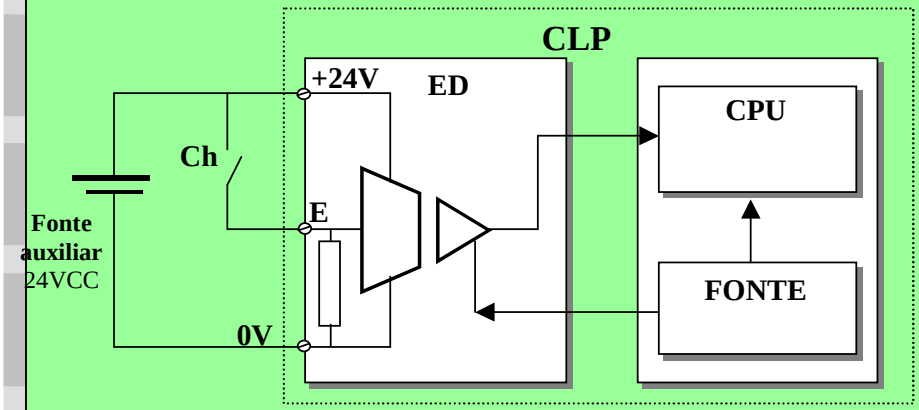
Módulo de comunicação serial:

Permite que o CLP se comunique serialmente com outros CLPs, com um computador encarregado de supervisionar todo o processo ou com sensores e atuadores de campo. Neste último caso apenas um par de fios torna-se capaz de captar as informações do processo e enviar as decisões de comando aos atuadores, proporcionando uma substancial simplificação na cablagem do sistema.

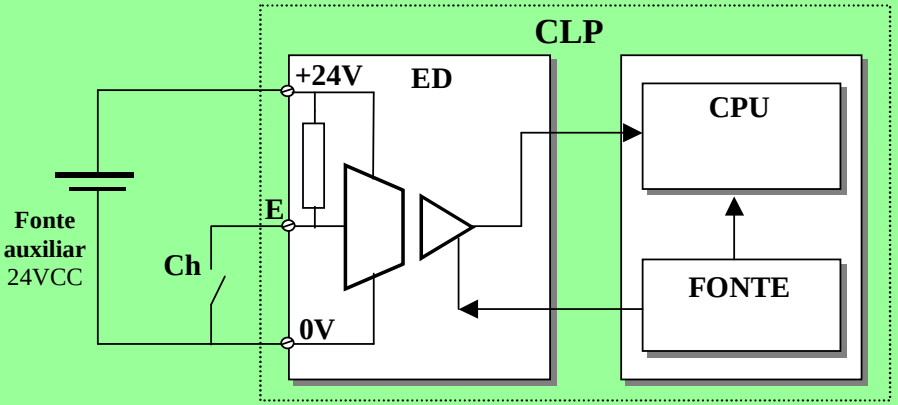
Entrada Digital (ED)

Permite que sinais binários, do tipo "falso-verdadeiro" sejam armazenados pelo CLP. Alguns exemplos de dispositivos usualmente conectados às EDs são: botoeiras, chaves fim de curso, contatos de relés, pressostatos, termostatos, relés de proteção, cortinas de luz, sensores de presença, sensores de proximidade, encoders, etc. As entradas digitais dividem-se em duas categorias, as de corrente continua com nível de tensão de 24V e as de corrente alternada com opções de 110V ou 220V.

Entradas do tipo CC:

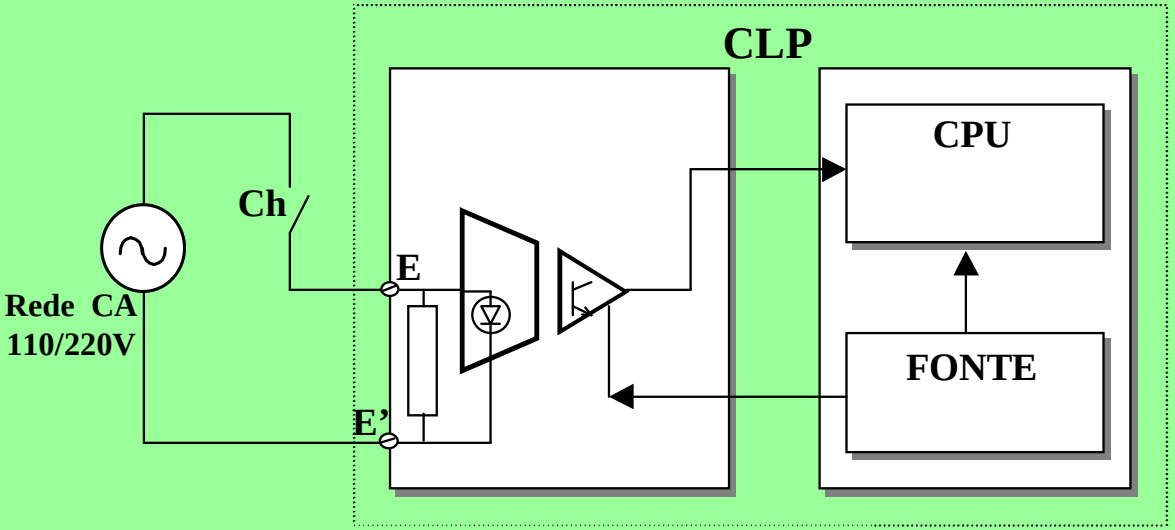


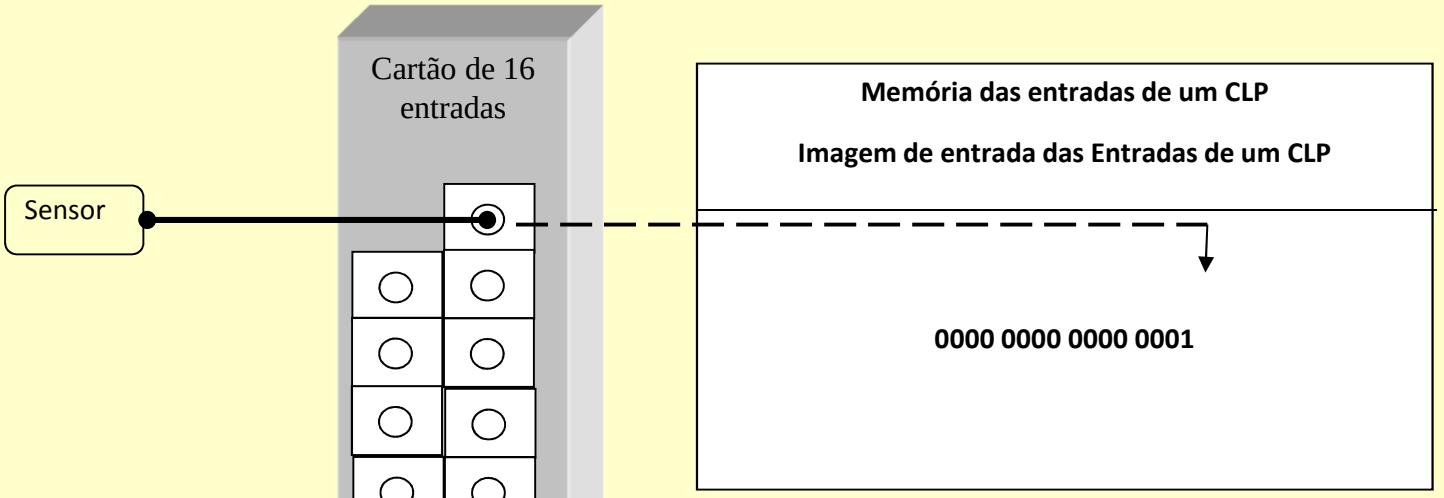
7a. ED positiva (sink)



7b. ED negativa (source)

Entrada Digital do tipo CA

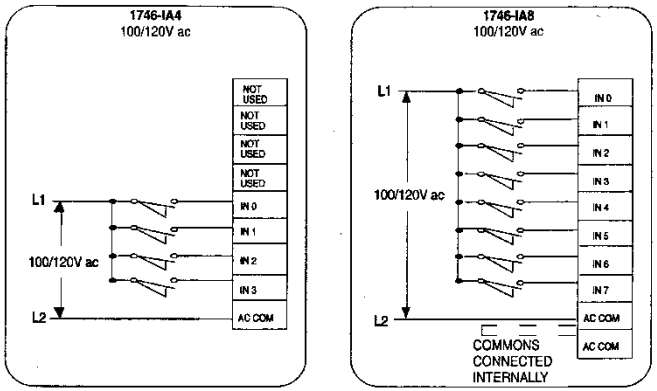


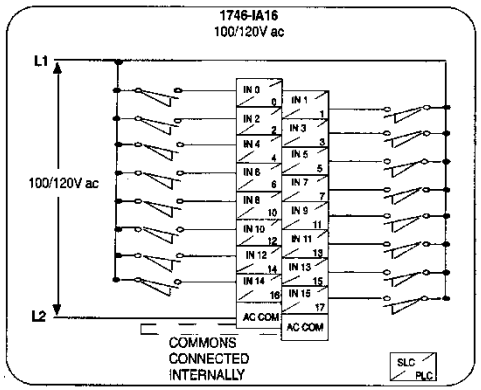


(Obs. Exemplo com Word de 16 bits)

Input Modules - ac

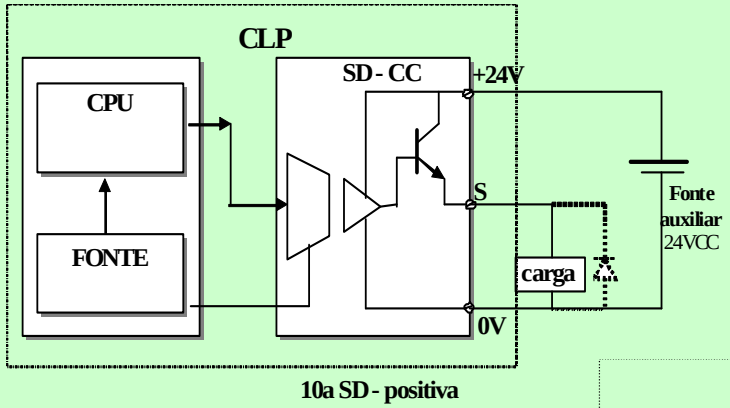
Figure 7: Wiring Diagrams (1746-IA4, -IA8, -IA16)



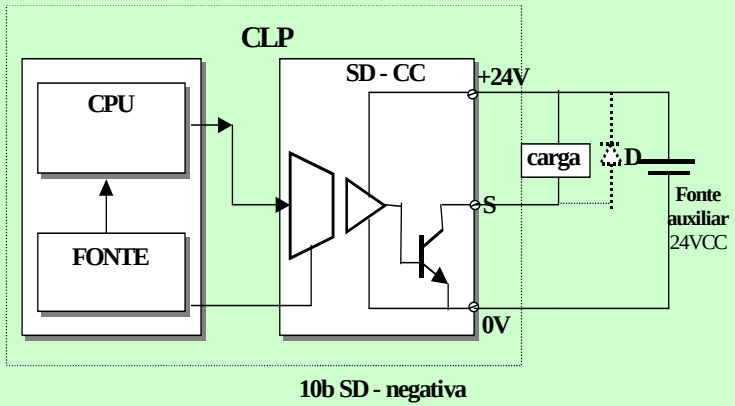


Saídas Digitais

Basicamente são três as categorias de saídas digitais: corrente contínua, corrente alternada e a relé. As figuras 10a,10b, 11 e 12 ilustram respectivamente: saída digital CC(positiva), saída digital CC(negativa), saída digital corrente alternada e saída digital a rele.



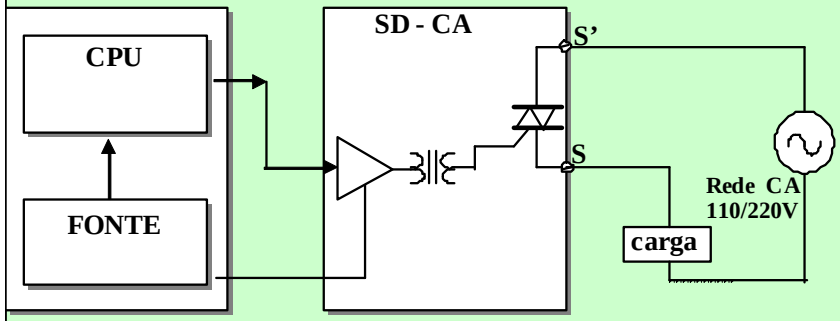
10a SD - positiva



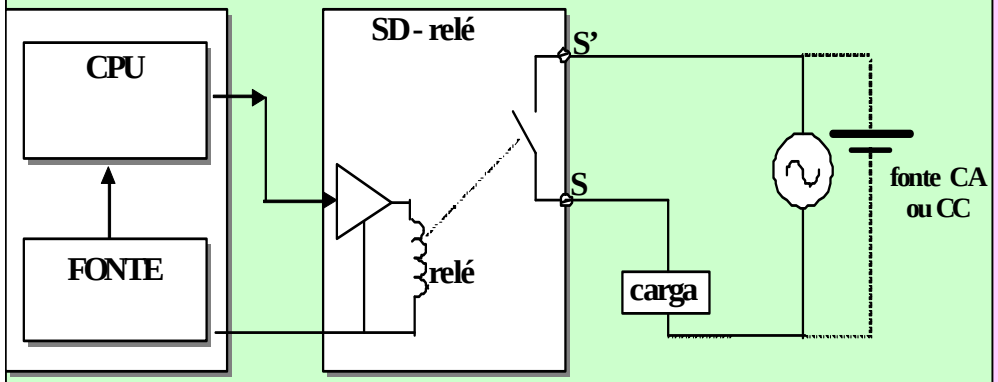
10b SD - negativa

Saídas do tipo CC

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



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



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

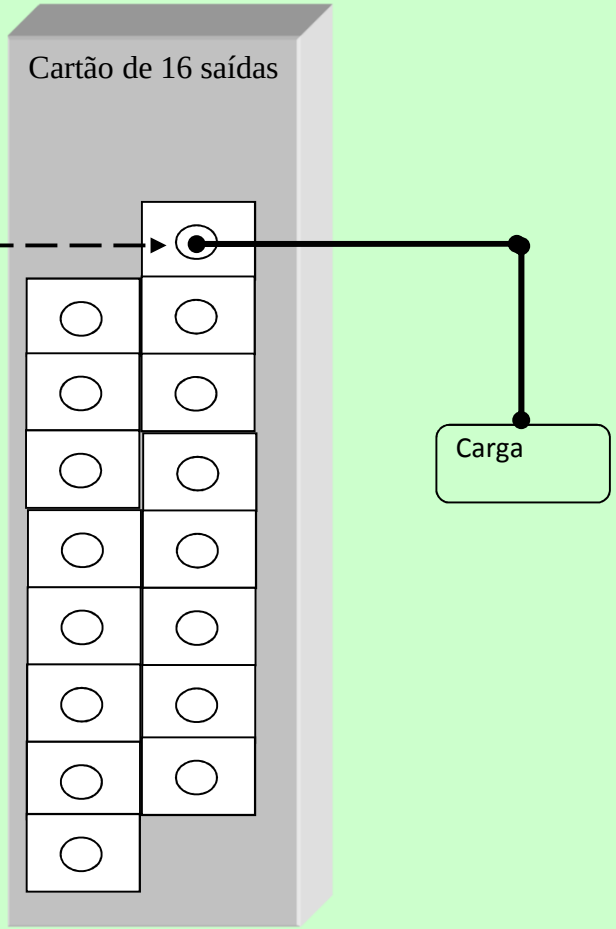
Saídas digital CA

Memória das saídas de um CLP

Imagem de entrada das Saídas de um CLP

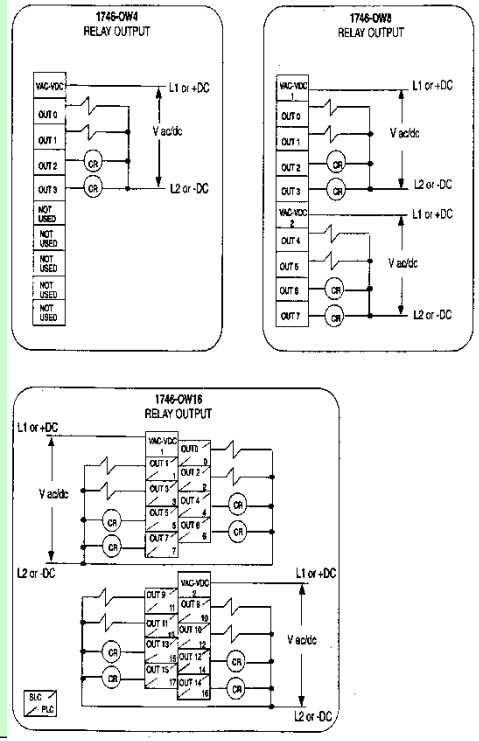
0000 0000 0000 0001

(Obs. Exemplo com Word de 16 bits)



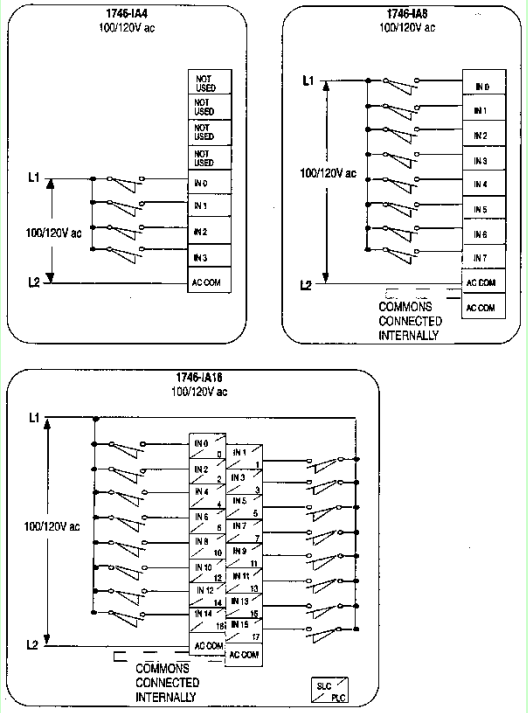
Relay Contact Output Modules

Figure 19: Wiring Diagrams (1746 -OW4, -OW8, -OW16)



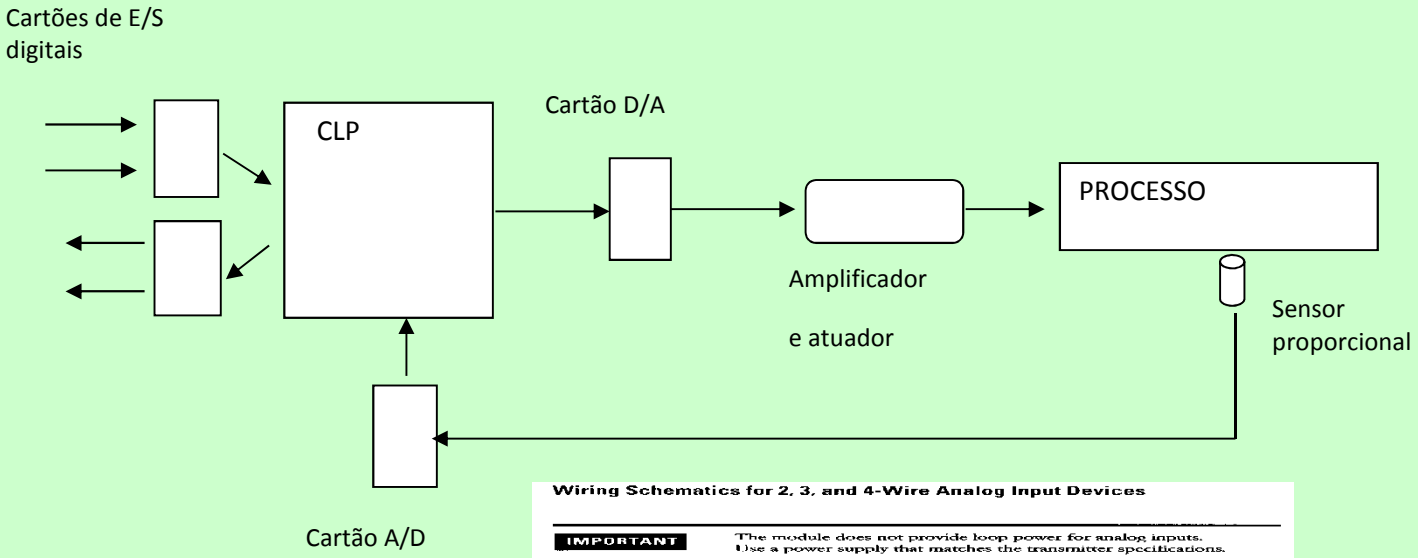
Input Modules - ac

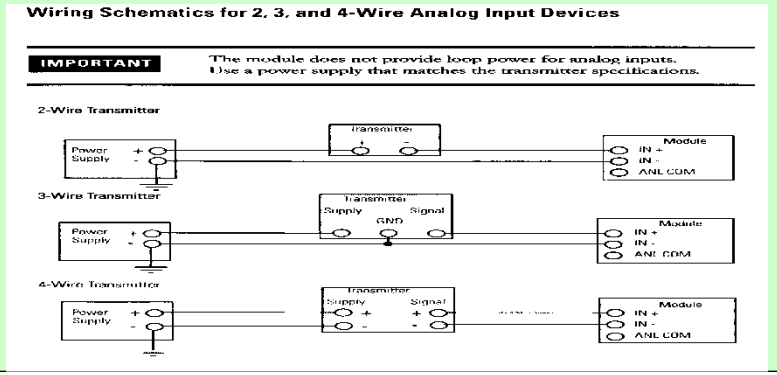
Figure 7: Wiring Diagrams (1746-IA4, -IA8, -IA16)



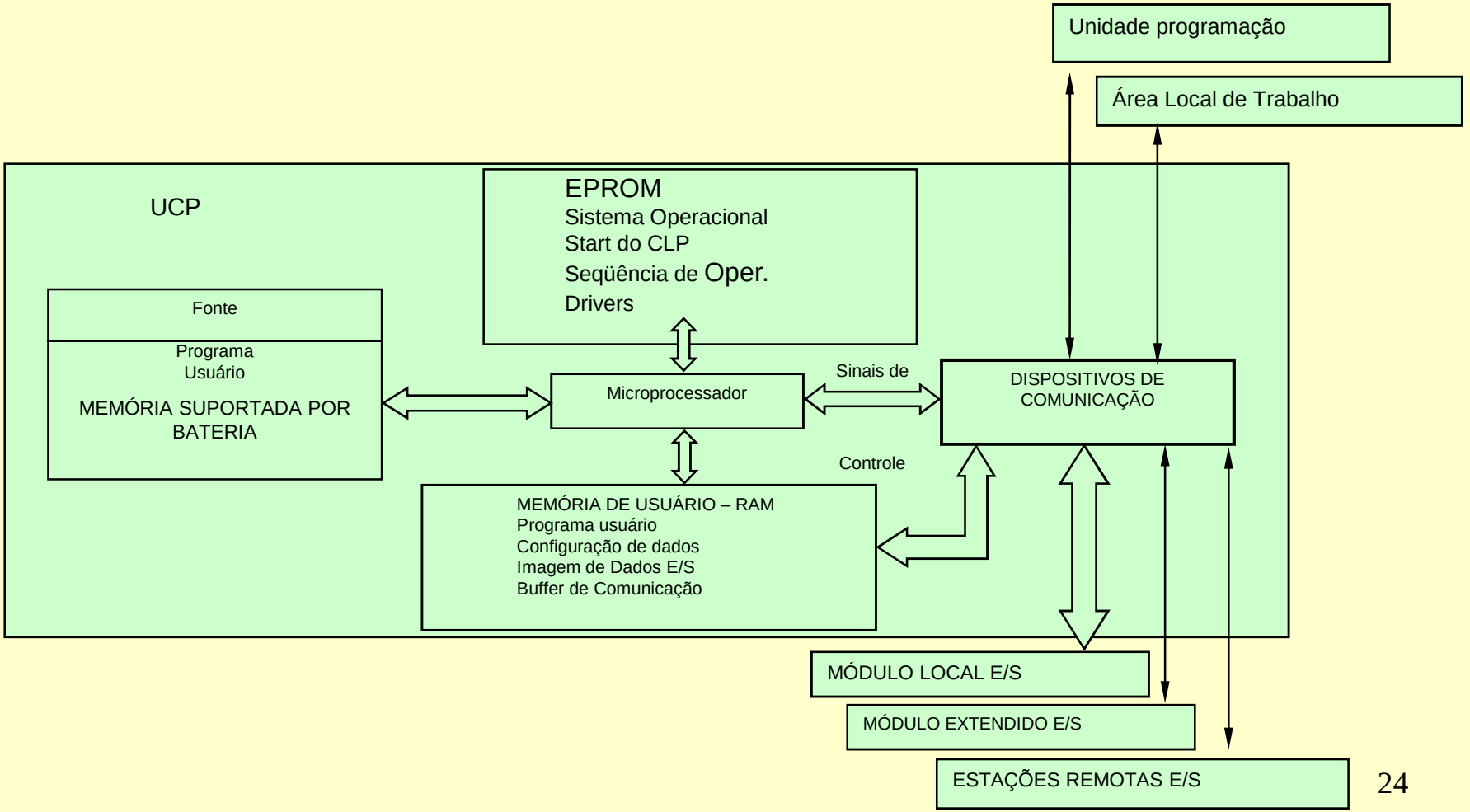
Entradas e Saídas Analógicas

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



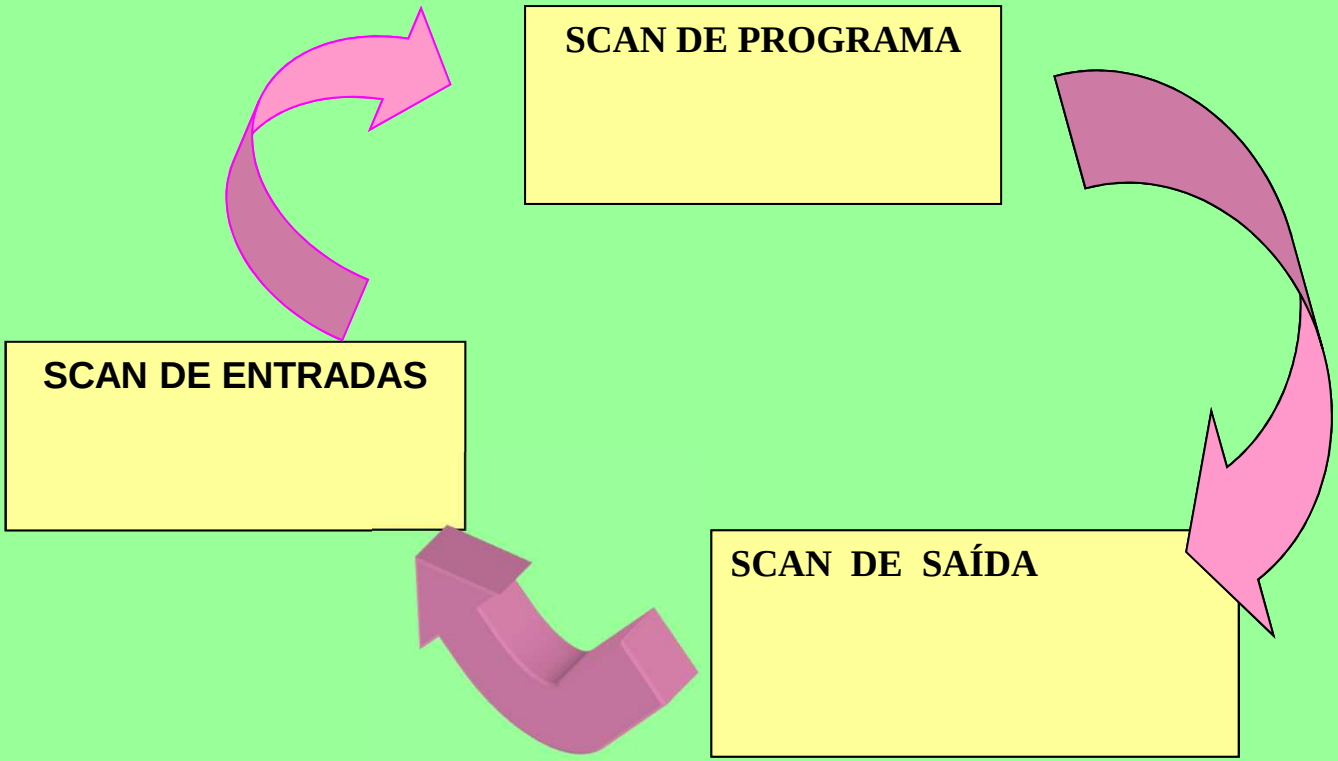


UPC - A Unidade Central de Processamento é a unidade que executa o programa de controle. Os CLPs tradicionais possuem uma UCP que realiza todas as funções de controle, leitura e escrita na memória. Também existem atualmente CLP que possuem duas UPCs operando com divisão de tarefas.



PRINCÍPIO DE FUNCIONAMENTO DE UM CLP GENÉRICO

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



O tempo de SCAN total depende da velocidade do CLP e do tamanho do programa de aplicação do usuário. Neste tempo deve ser computado o tempo de atualização dos módulos de entrada e saída. Determinadas instruções dos CLPs por serem mais complexas necessitam de um tempo maior de processamento. Os valores típicos de SCANs dos CLPs comerciais variam de 1 ms até 10 ms para cada 1 k típico de instruções. Portanto deve conhecer antecipadamente qual o período de ciclo do processo que se deseja controlar utilizando um CLP, e se o tempo do SCAN do CLP é significativamente menor para se atingir as condições básicas de controlabilidade.

LINGUAGENS DE PROGRAMAÇÃO DE CLPs

Existe hoje no mercado uma grande variedade de linguagens para programação de CLPs.
Elas são:

a - “Ladder Diagrams”.

b -“Functional Blocks”.

c - “Boolean Mnemonics”.

d - “English Statements”.

e – “GRAFSET” ou Sequential Function Chart (SFC).

Linguagens de Programação de Controladores Programáveis IEC 61131-3

O IEC - International Electrotechnical Commission é o responsável pela padronização dessas linguagens de programação, sendo a norma IEC 61131-3 PLC Programming Languages a recomendada para o assunto em questão. A norma IEC 61131-3 foi publicada em 1994.

A norma IEC 61131 consiste de 8 partes. As principais são:

1. **Informações gerais;**
2. **Requisitos de equipamentos e testes;**
3. **Linguagens de programação de CLP;**
4. **Diretrizes do usuário;**
5. **Comunicações.**

Linguagens de Programação de Controladores Programáveis – IEC 61131-3

Parte	Título	Conteúdo	Publicação
Part 1	*General Information*	Definição da terminologia e conceitos	2003 (2ª ed.)
Part 2	*Equipment Requirements and Tests*	Testes de verificação e fabricação eletrônica e mecânica	2003 (2ª ed.)
Part 3	***Programming Languages***	**Estrutura do software do Controlador Programável (CP), linguagens e execução de programas**	**2003 (2ª ed.)**
Part 4	*User Guidelines*	Orientações para seleção, instalação e manutenção de CPs	2004 (2ª ed.)
Part 5	*Communications*	Funcionalidades para comunicação com outros dispositivos	2000 (1ª ed.)
Part 6	*Reservada*		
Part 7	*Fuzzy Control Programming*	Funcionalidades de software, incluindo blocos funcionais padrões para tratamento de lógica nebulosa dentro de CPs	2000 (1ª ed.)
Part 8	*Guidelines for the Application and Implementation of Programming Languages*	Orientações para implementação das linguagen s IEC 61131-3	2003 (2ª ed.)

Modelo Funcional de um CLP

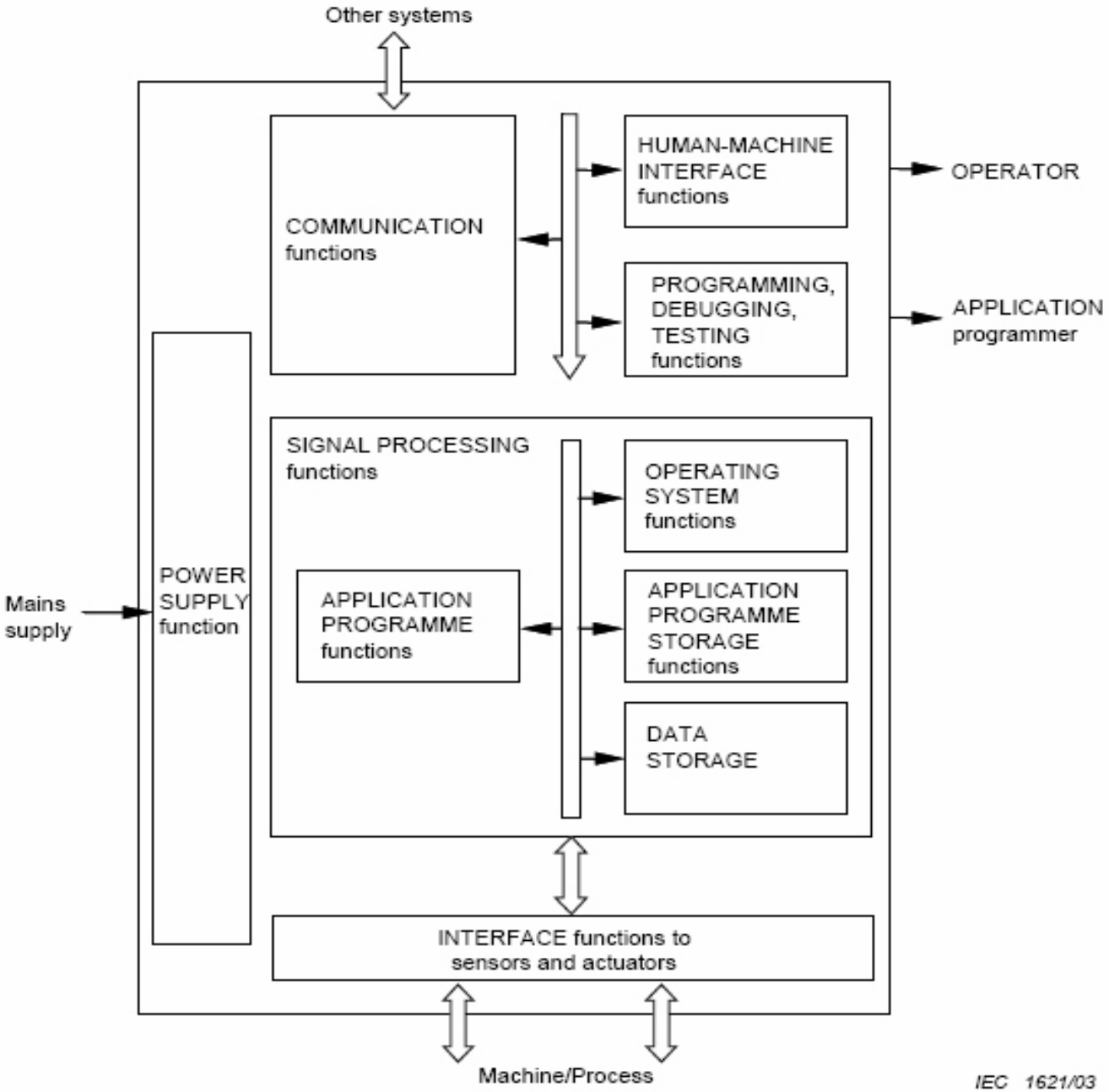


Figure 1 – Basic functional structure of a PLC-system

Linguagens de Programação de Controladores Programáveis – IEC 61131-3

Modelo de hardware de um CLP segundo a IEC 61131-5

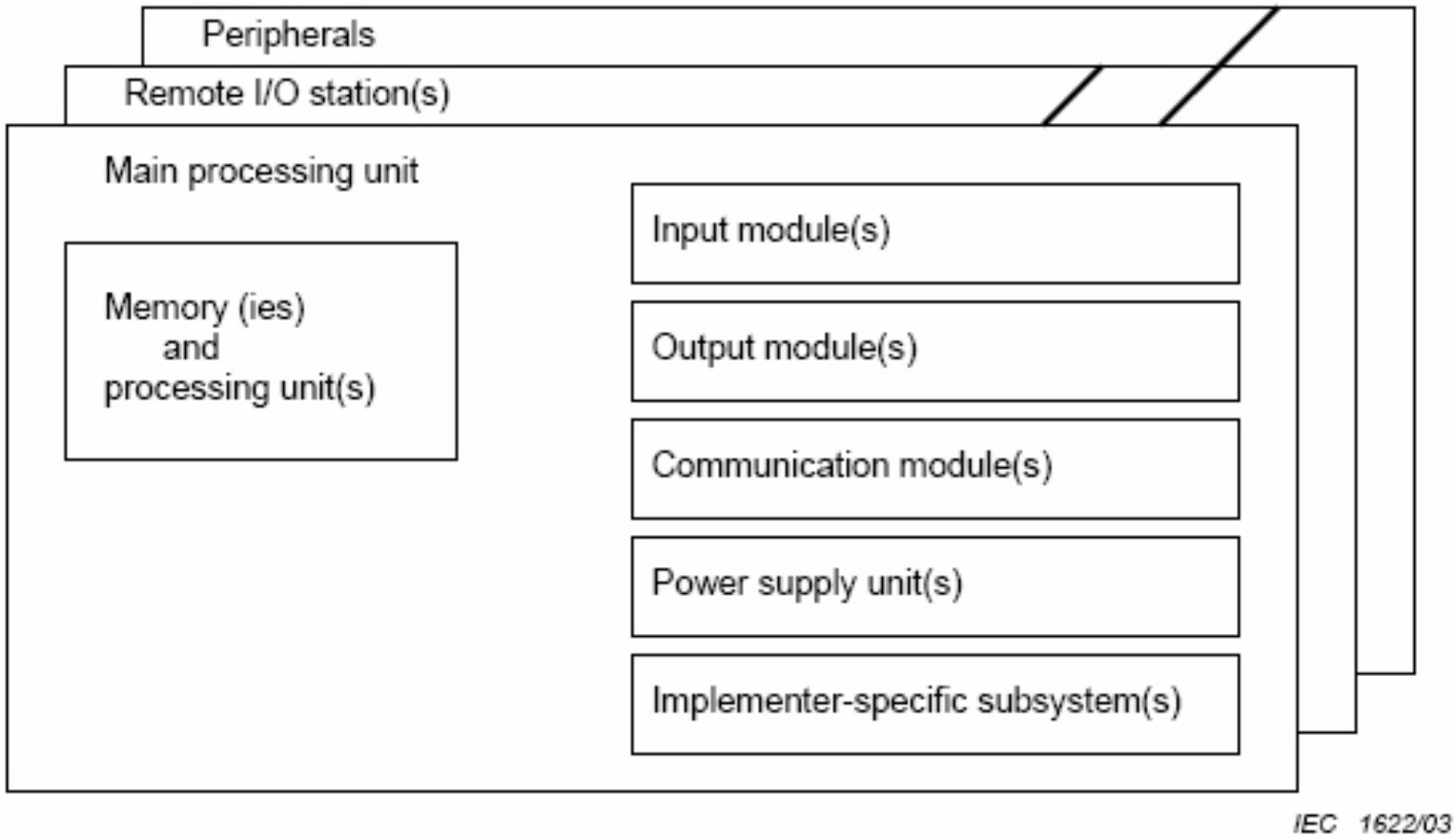


Figure 2 – Programmable controller hardware model (from IEC 61131-5)

Linguagens de Programação de Controladores Programáveis: Linguagens – IEC 61131-3

As características inovadoras da norma são:

1) A norma encoraja o desenvolvimento de um programa bem estruturado “top-down” ou “bottom-up”;

2) Ela exige que exista uma definição forte de tipo de variável;

3) Suporta controle completo da execução do programa através do controle da execução de diferentes partes do programa com diferentes eventos e taxas de varredura;

4) Suporte completo para a descrição de um processo com comportamento sequencial complexo através da linguagem gráfica SFC (Sequencial Function Chart);

5) Seleção da linguagem mais adequada ao processo;

6) Software de programação independente do fornecedor do equipamento.

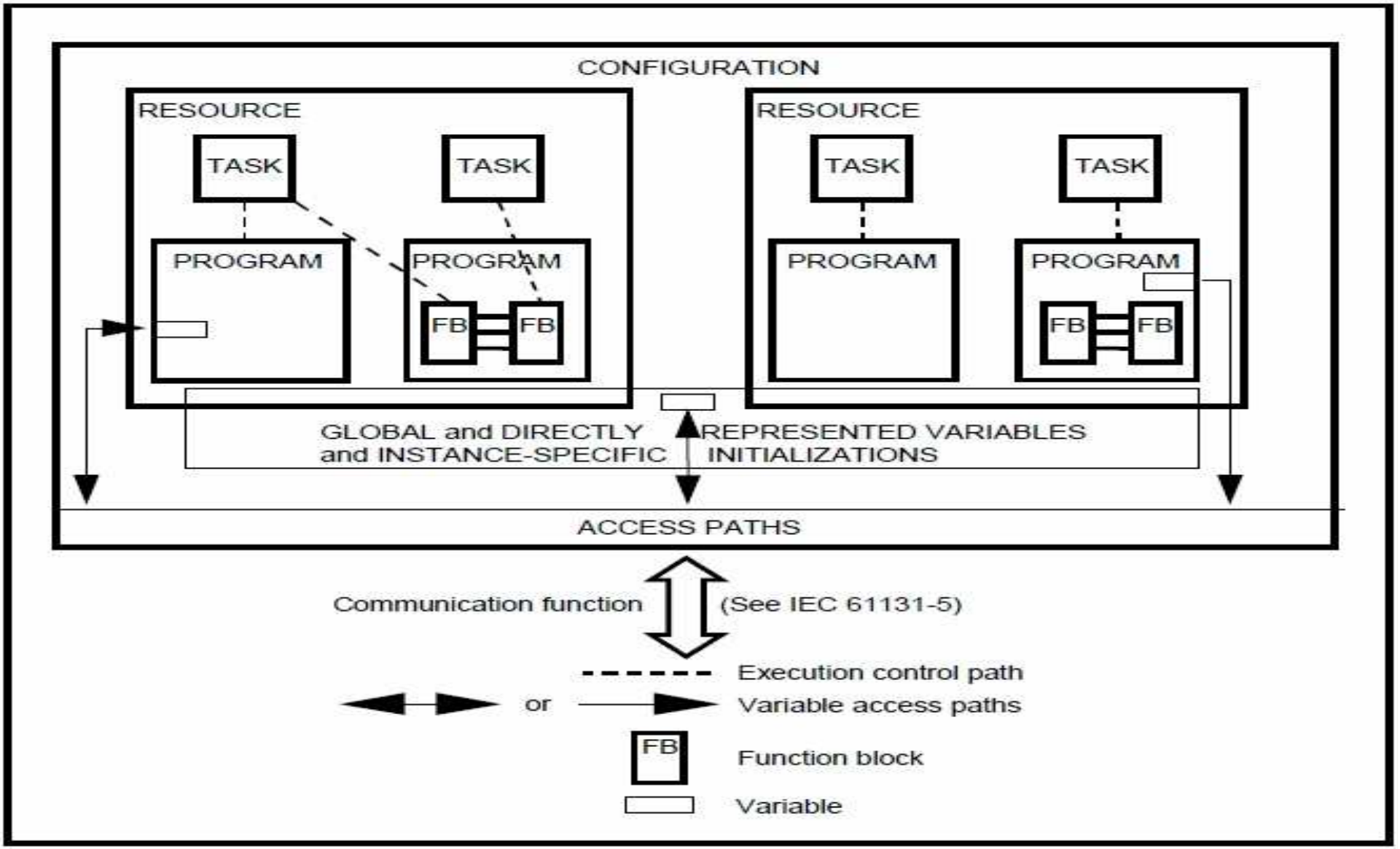
Definições da norma IEC 61131-3

A **norma IEC 61131-3 pode ser dividida em duas partes:**

- **Elementos Comuns – tipos de dados, variáveis, constantes, configuração, recursos e tarefas;**
- **Linguagens de Programação – cinco linguagens de programação.**

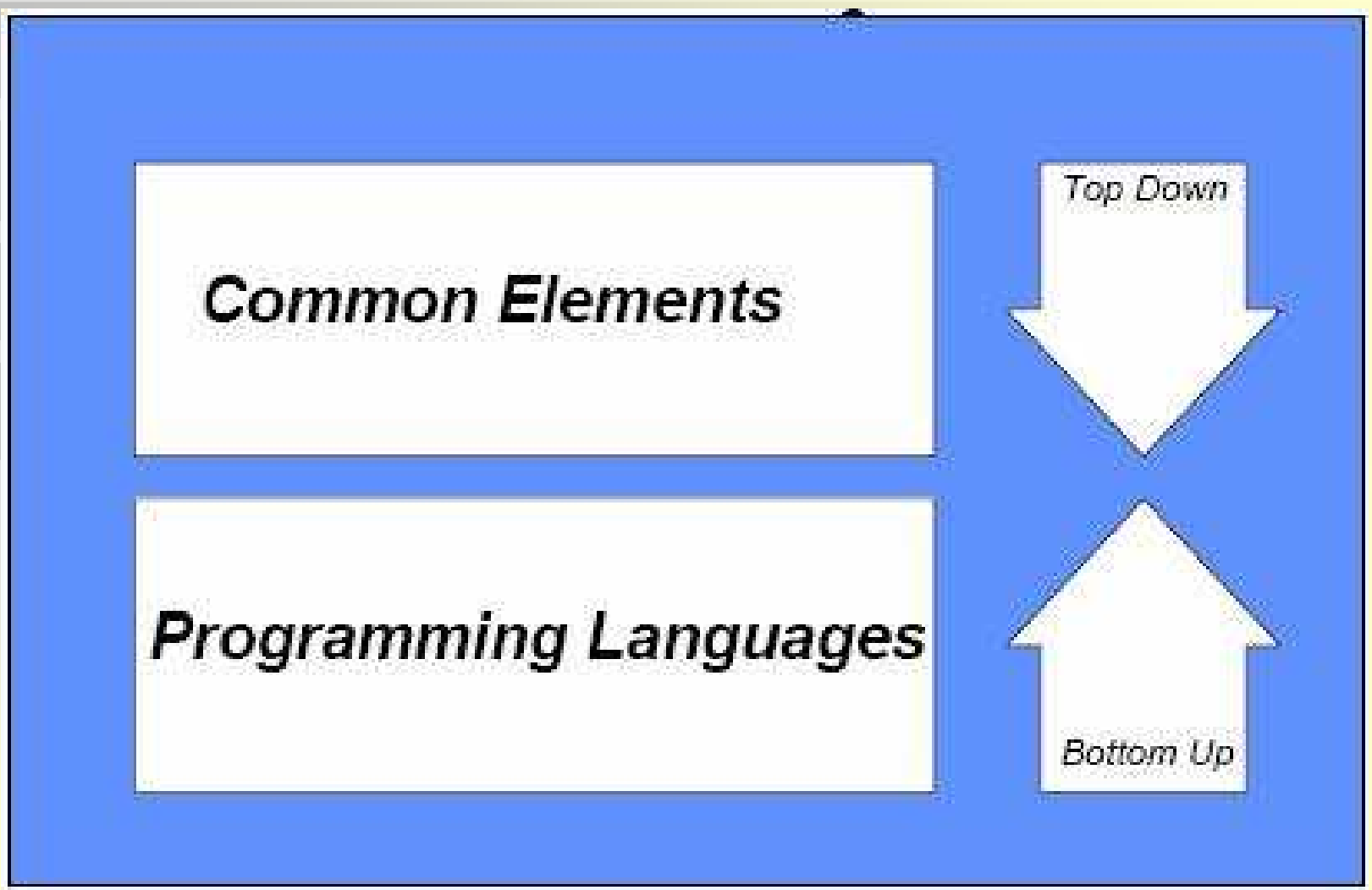
Definições da norma IEC 61131-3 – Elementos Comuns

Configuração, recursos e tarefas.



Definições da norma IEC 61131-3

Estruturação **Top-down** X **Bottom-up**



Desenvolvimento do programa

Os estágios para o desenvolvimento de **um programa são:**

- **Escrever as instruções;**
- **Editar o programa;**
- **Verificar e corrigir erros de sintaxe;**
- **Carregá-lo e testá-lo no controlador.**

Classificação das linguagens de programação segundo a norma IEC 61131-3

Classes	Linguagens
Textuais	IL (Instruction List)
	ST (Structured Text)
Gráficas	LD (Ladder Diagram)
	FBD (Function Block Diagram)
	SFC (Sequential Function Chart)

IL

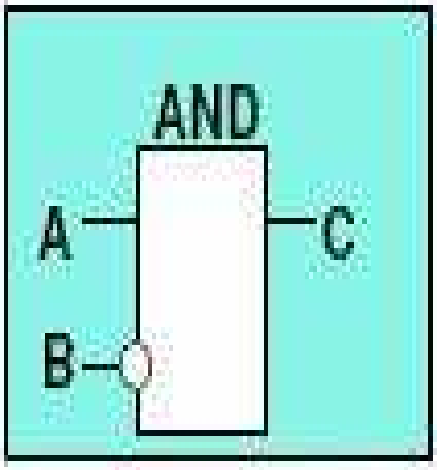
```
LD    A
ANDN  B
ST    C
```

ST

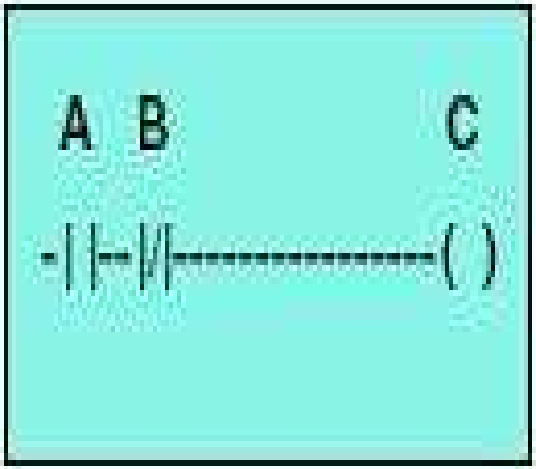
```
C:= A  AND  NOT B
```

SFC

FBD



LD



Ladder Diagram (LD)

Utilizar Linguagem Ladder quando o processo possui:

- Execução contínua ou paralela de múltiplas operações não sequenciais;
- Operações baseadas em bit ou em lógica booleana;
- Operações lógicas complexas;
- Processamento de mensagem e comunicação;
- Intertravamento de máquinas;
- Operações que o pessoal de manutenção deverá interpretar para resolver problemas em máquinas ou processos.

Function Block Diagram (FBD)

Utilizar Linguagem bloco de funções quando o processo possui:

- Controle de drives e processos contínuos;
- Loop de controle.

Sequential Function Chart (SFC)

Utilizar Linguagem SFC quando o processo possui:

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

Structured Text (ST)

Utilizar Linguagem texto estruturado quando o processo possui:

- Operações matemáticas complexas;
- Processamento de protocolos ou manuseio de strings ASCII;
- Processamento de dados com loop´s em tabelas ou vetores de dados.

- Ambiente de programação;
- Controlador (Hardware) e Aplicativo (Software);
- Conceitos de Download e Upload de aplicativos;
- Tipos de variáveis a serem utilizados na programação: variáveis simples ou estruturadas;
- Tipos e modo de alocação dos módulos de E/S;
- Estruturação hierárquica da programação em tarefas, programas e rotinas;
- Scan de execução do programa aplicativo;
- Condições a serem seguidas nas mudanças de modo auto/manual nos comandos e acionamentos.

Estruturação Aplicativo de Controladores Programávei

- Análise do processo a ser automatizado – utilização de fluxogramas, lista de E/S e descritivos, listas de comunicação;

- Conceitos de Núcleo de Processo e Interfaces (E/S, redes de comunicação, supervisório e outros);

- Conceitos de estruturação Top-Down e Down-Top;

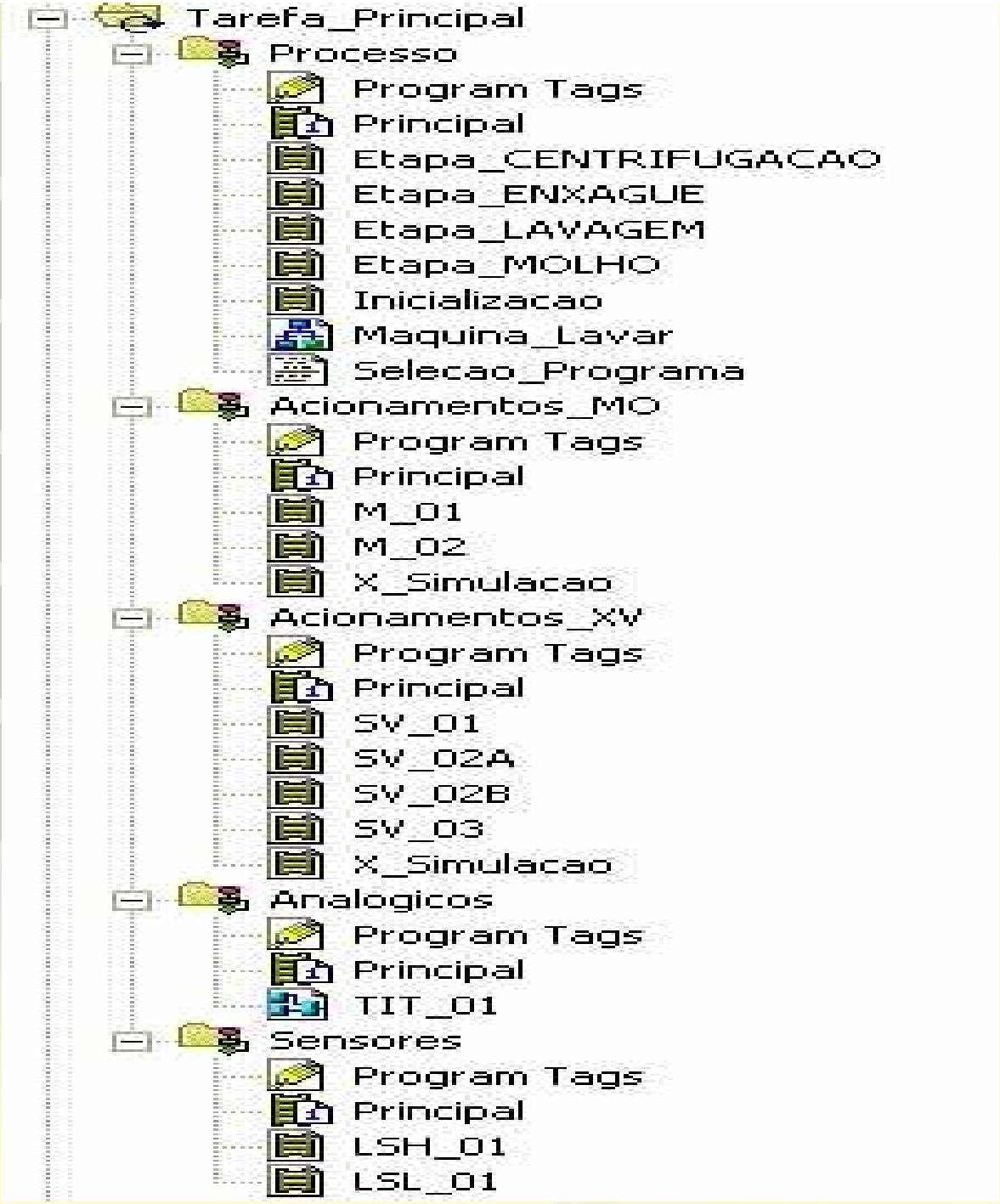
- Requisitos a serem atendidos – facilidade de entendimento do processo, facilidade de manutenção e outros;

- Estruturação da Base de Dados;

- Estruturação das Tarefas/Programas/Rotinas;

- Utilização de Rotinas especiais (Periódicas e com Prioridade) para malhas de controle, intertravamentos críticos e totalizações.

Estruturação de PROGRAMAS



Estruturação de Base de DADOS

Name	Data Type	Description
M_01	MOTOR	Motor M-01 (Motor para Lavagem)
M_01.CMD_LIGA_DIR	BOOL	Motor M-01 (Motor para Lavagem) - Comando Lógico de Partida Direta
M_01.CMD_LIGA_REV	BOOL	Motor M-01 (Motor para Lavagem) - Comando Lógico de Partida Reversa
M_01.CMD_MANUAL	M_CMD_EXTERNO	Motor M-01 (Motor para Lavagem) - Comando do Motor via Supervisório
M_01.CMD_MANUAL.LIGA_DIR	BOOL	Motor M-01 (Motor para Lavagem) - Comando Liga Direto Supervisório
M_01.CMD_MANUAL.LIGA_REV	BOOL	Motor M-01 (Motor para Lavagem) - Comando Liga Reverso Supervisório
M_01.CMD_MANUAL.DESLIGA	BOOL	Motor M-01 (Motor para Lavagem) - Comando Desliga Supervisório
M_01.CMD_MANUAL.MAN_AUTO	BOOL	Motor M-01 (Motor para Lavagem) - 1=Manual / 0=Auto
M_01.CMD_MANUAL.BLOQUEIO	BOOL	Motor M-01 (Motor para Lavagem) - Bloqueio de Acionamento
M_01.CMD_MANUAL.RESET	BOOL	Motor M-01 (Motor para Lavagem) - Reset de Falha
M_01.CMD_AUTO	M_CMD_INTERNO	Motor M-01 (Motor para Lavagem) - Comando do Motor via CLP
M_01.CMD_AUTO.LIGA_DIR	BOOL	Motor M-01 (Motor para Lavagem) - Comando Liga Direto CLP
M_01.CMD_AUTO.LIGA_REV	BOOL	Motor M-01 (Motor para Lavagem) - Comando Liga Reverso CLP
M_01.CMD_AUTO.DESLIGA	BOOL	Motor M-01 (Motor para Lavagem) - Comando Desliga CLP
M_01.STS	M_STS_GERAL	Motor M-01 (Motor para Lavagem) - Status do Motor
M_01.STS.FALHA	BOOL	Motor M-01 (Motor para Lavagem) - Falha do Motor
M_01.STS.LIGADO_DIR	BOOL	Motor M-01 (Motor para Lavagem) - Retorno de Ligado Direto
M_01.STS.LIGADO_REV	BOOL	Motor M-01 (Motor para Lavagem) - Retorno de Ligado Reverso
M_01.STS.INTLCK	BOOL	Motor M-01 (Motor para Lavagem) - Intertravamento
M_01.CORRENTE	REAL	Motor M-01 (Motor para Lavagem) - Corrente do Motor
M_01.TIMER_FALHA	TIMER	Motor M-01 (Motor para Lavagem) - Temporizador de Falha

Estruturação de Base de DADOS

Name	Data Type	Description
– SV_01	XV	Válvula SV-01 (Válvula de Entrada de Água)
SV_01.CMD_ACIONA	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Aciona Válvula
– SV_01.CMD_MANUAL	XV_CMD_EXTERNO	Válvula SV-01 (Válvula de Entrada de Água) - Comando da Válvula via Supervisório
SV_01.CMD_MANUAL.ABRE	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Abre Válvula via Supervisório
SV_01.CMD_MANUAL.FECHA	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Fecha Válvula via Supervisório
SV_01.CMD_MANUAL.MAN_AUTO	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - 1=Manual / 0=Auto
SV_01.CMD_MANUAL.BLOQUEIO	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Bloqueio de Acionamento
SV_01.CMD_MANUAL.RESET	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Reset de Falha
– SV_01.CMD_AUTO	XV_CMD_INTERNO	Válvula SV-01 (Válvula de Entrada de Água) - Comando da Válvula via CLP
SV_01.CMD_AUTO.ABRE	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Abre Válvula via CLP
SV_01.CMD_AUTO.FECHA	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Fecha Valvula via CLP
– SV_01.STS	XV_STS_GERAL	Válvula SV-01 (Válvula de Entrada de Água) - Status da Válvula
SV_01.STS.FALHA	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Falha da Válvula
SV_01.STS.ABERTO	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Retorno de Válvula Aberta
SV_01.STS.FECHADO	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Retorno de Válvula Fechada
SV_01.STS.INTLCK	BOOL	Válvula SV-01 (Válvula de Entrada de Água) - Interlock da Válvula
+ SV_01.TIMER_FALHA	TIMER	Válvula SV-01 (Válvula de Entrada de Água) - Timer de Falha

INTRODUÇÃO À LINGUAGEM LADDER

INSTRUÇÕES BÁSICAS EM DIAGRAMA LADDER (CLPs)

A linguagem de programação Ladder é um sistema gráfico de símbolos e termos que evoluiu dos diagramas Ladder elétricos, que representam a maneira como a corrente elétrica circula pelos dispositivos, de forma a completar um circuito elétrico.

Basicamente um programa no CLP se divide em instruções de entrada e em instruções de saída

Instruções de entrada

Instruções de saída

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

- **lógica de rele ou instrução de Bit,**
- **temporização e contagem,**
- **aritméticas,**
- **manipulação de dados,**
- **controle de fluxo,**
- **transferência de dados,**
- **avançadas.**

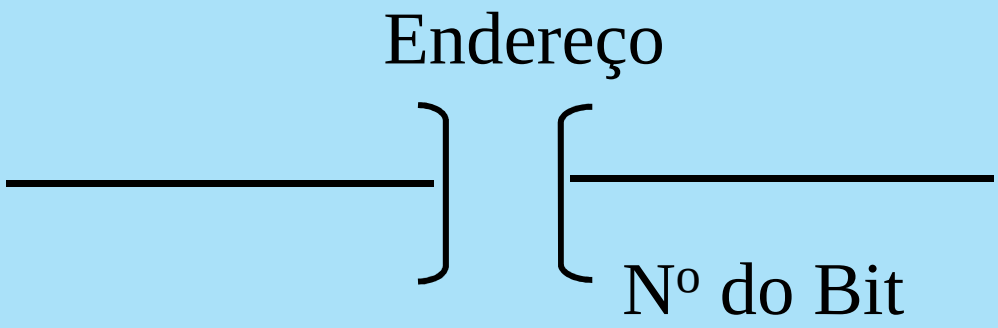
Uma instrução de Bit pode ser de entrada ou de saída.

Durante a execução de uma instrução de entrada o estado de um Bit em um determinado endereço da memória RAM do CLP é examinado.

Durante a execução de uma instrução de saída de bit o estado de um bit de um determinado endereço da memória RAM do CLP é alterado para 0 ou para 1 conforme haja ou não continuidade lógica da linha que a instrução está relacionada.

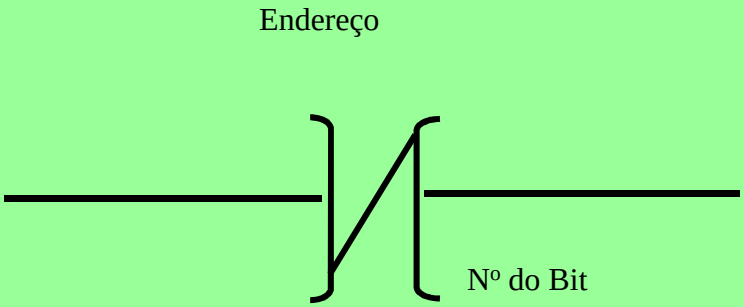
Lógica de relé ou instrução de Bit

INSTRUÇÃO (XIC) → Examinar se energizado.



Estado do BIT	Instrução XIC
0	Falsa
1	Verdadeira

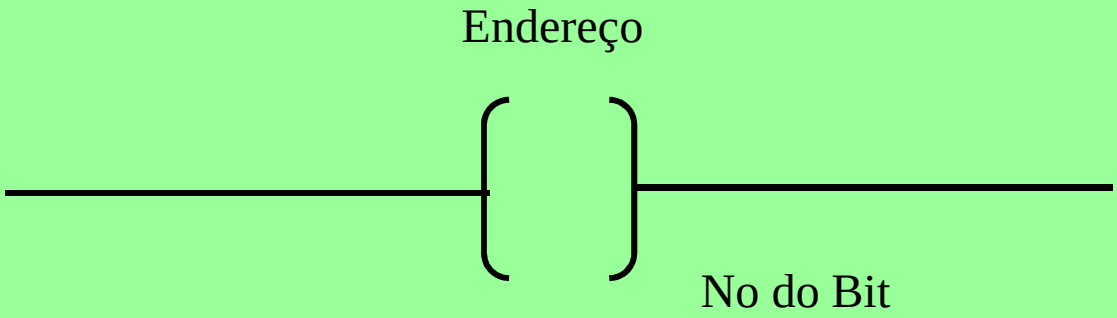
Instrução **(XIO) → Examinar se o Bit está com o valor “0”.**



Estado do BIT	Instrução XIO
0	Verdadeira
1	Falsa

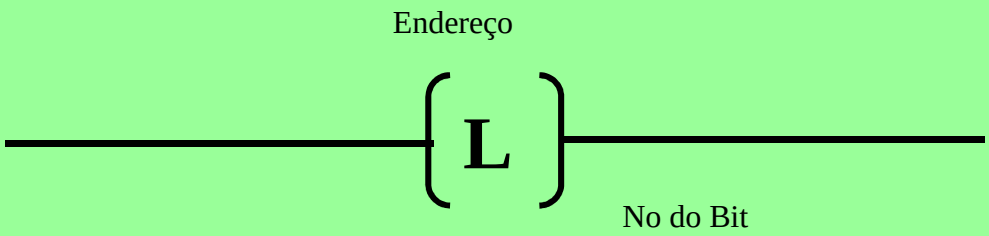
Instrução **(OTE) → Energizar saída**

Caso haja continuidade lógica da linha o bit endereçado pela instrução será colocado no estado lógico 1. Se não houver continuidade na linha o bit endereçado pela instrução será colocado no estado lógico 0.



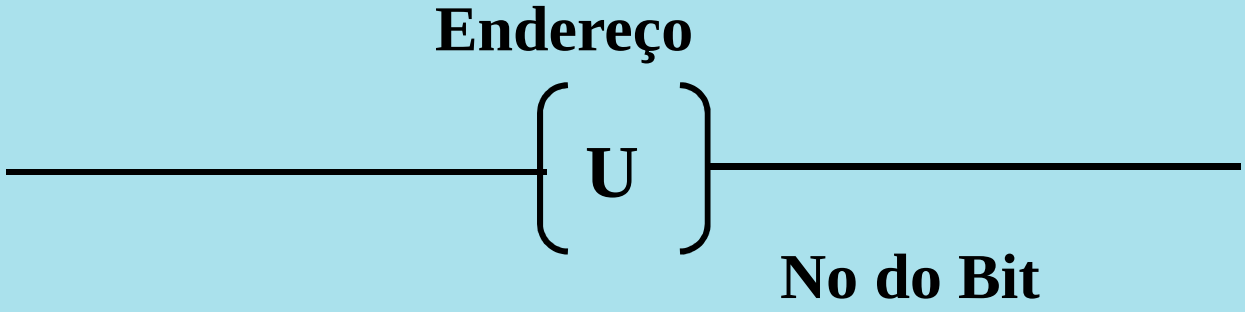
Instrução (OTL) → Energizar saída com retenção

Uma vez habilitada a saída endereçada pela instrução, a mesma somente será desabilitada caso a instrução OTU seja acionada.

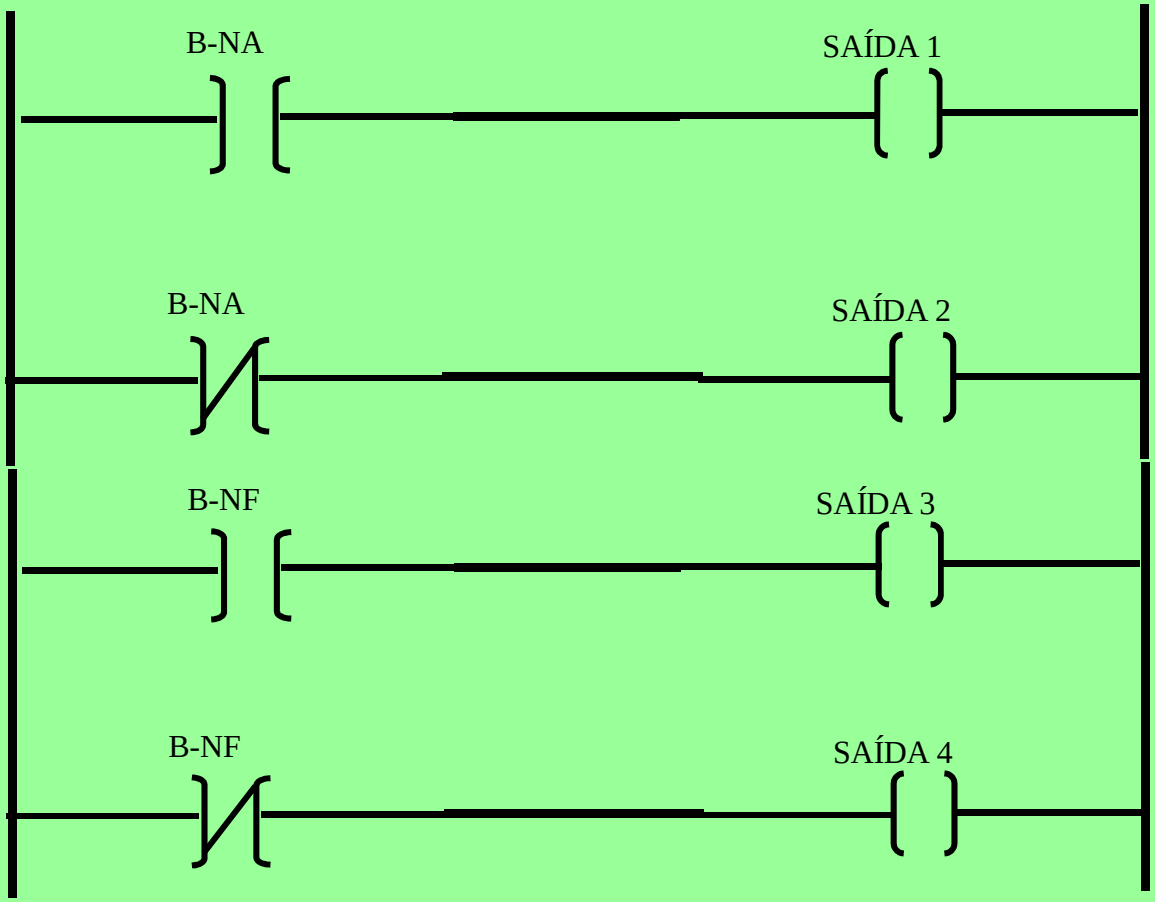


Instrução (OTU) → Desabilitar saída com retenção

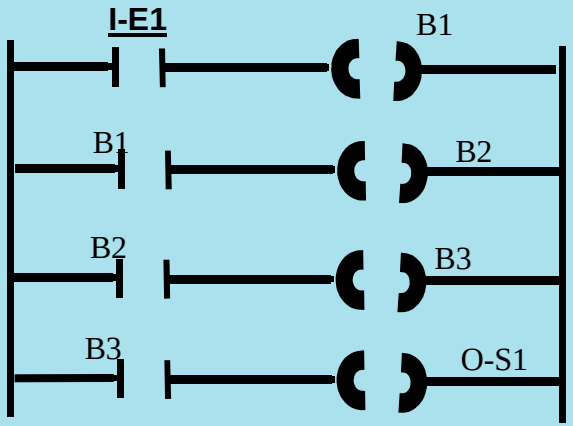
Desabilita uma saída habilitada por uma instrução OTL.



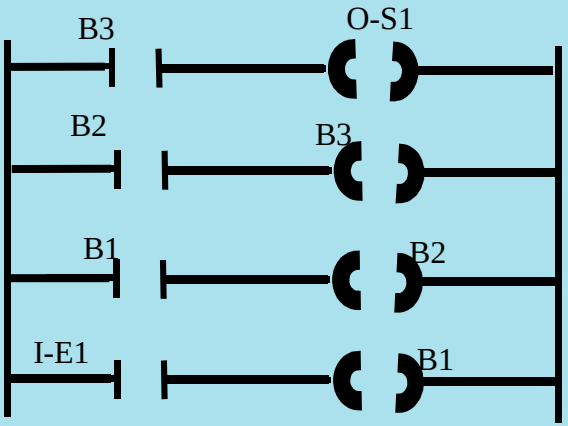
EXEMPLO 1



Exemplo 2 e 3



Se I-E1 for verdadeiro,
A Saída conectada à O-S1
Será acionada após um Scan.



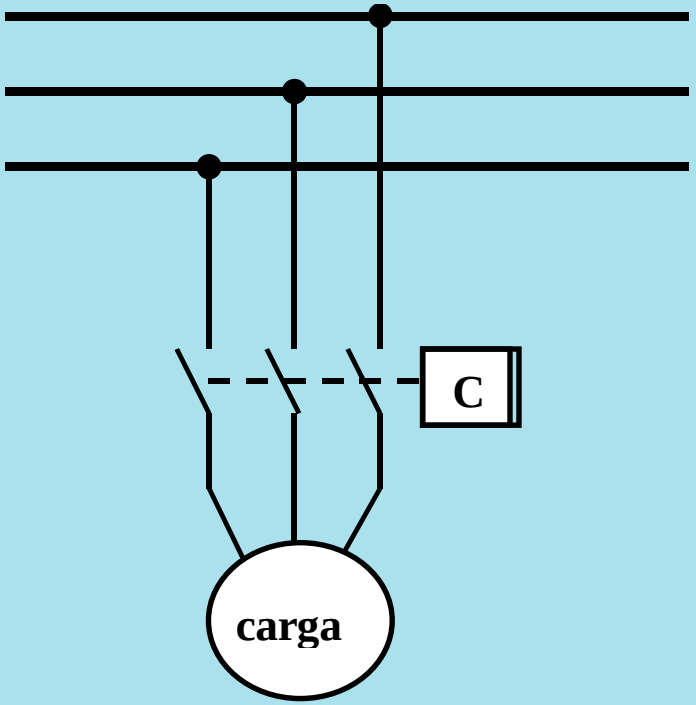
Se I-E1 for verdadeiro,
A Saída conectada à O-S1
Será acionada após quatro Scan.

Suponha que um circuito elétrico tradicional esteja comandando a partida de um de um motor trifásico. O circuito de controle emprega:

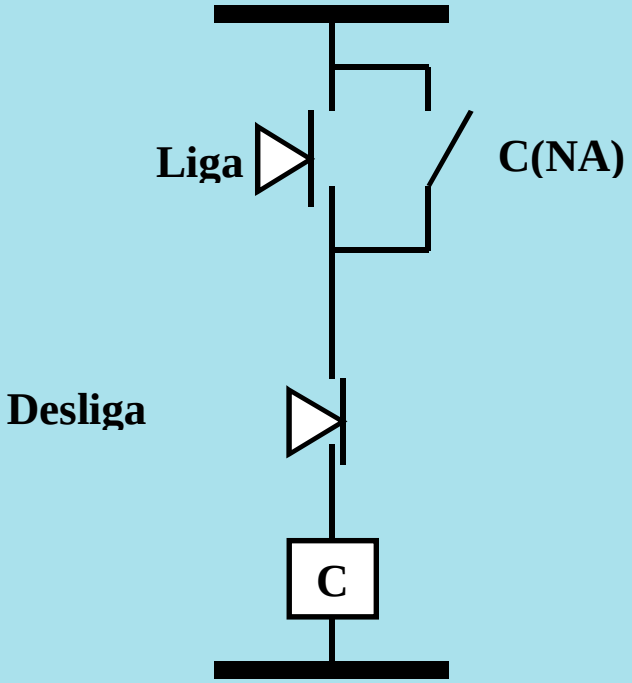
a - uma botoeira normalmente fechada,

b - uma botoeira normalmente aberta.

c - um contator para acionar o motor trifásico

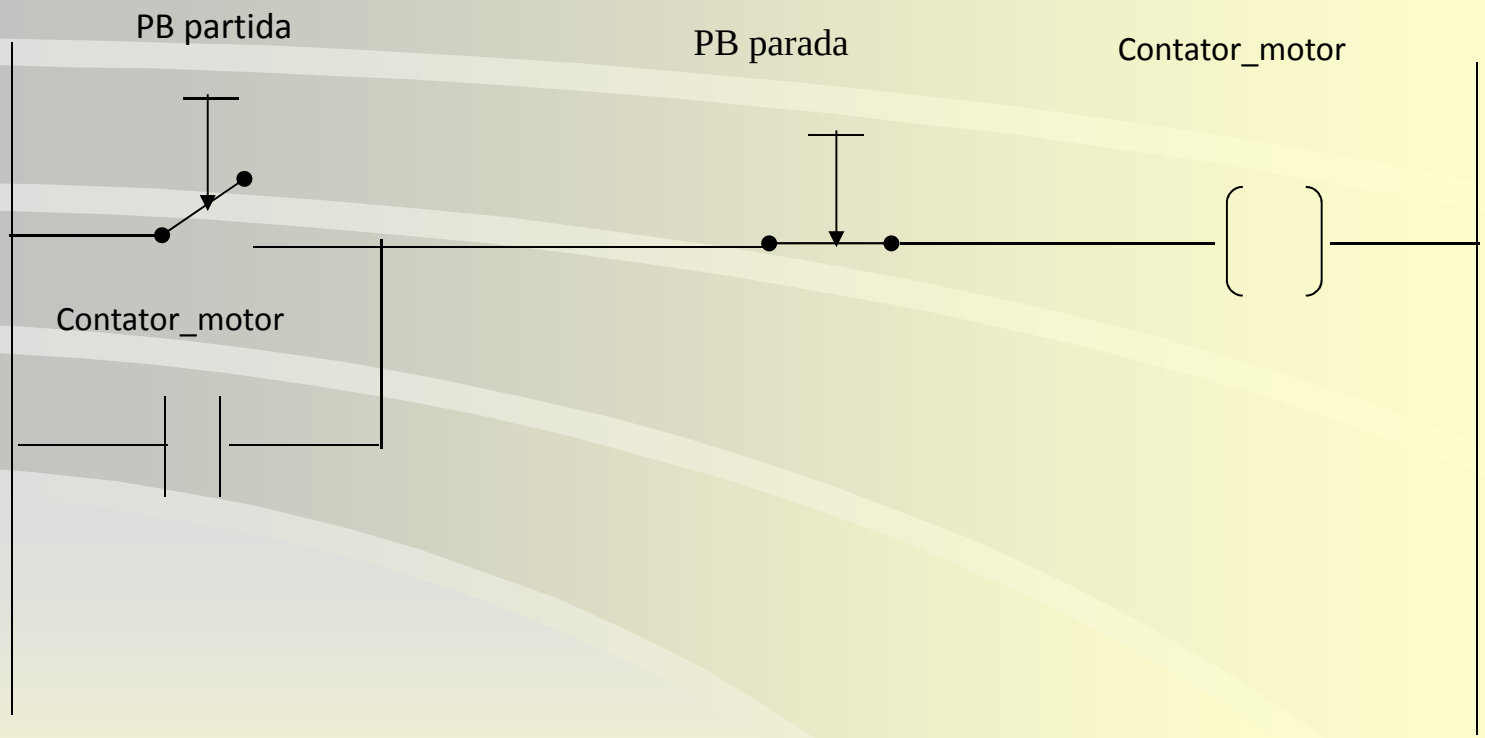


Circuito de Potência

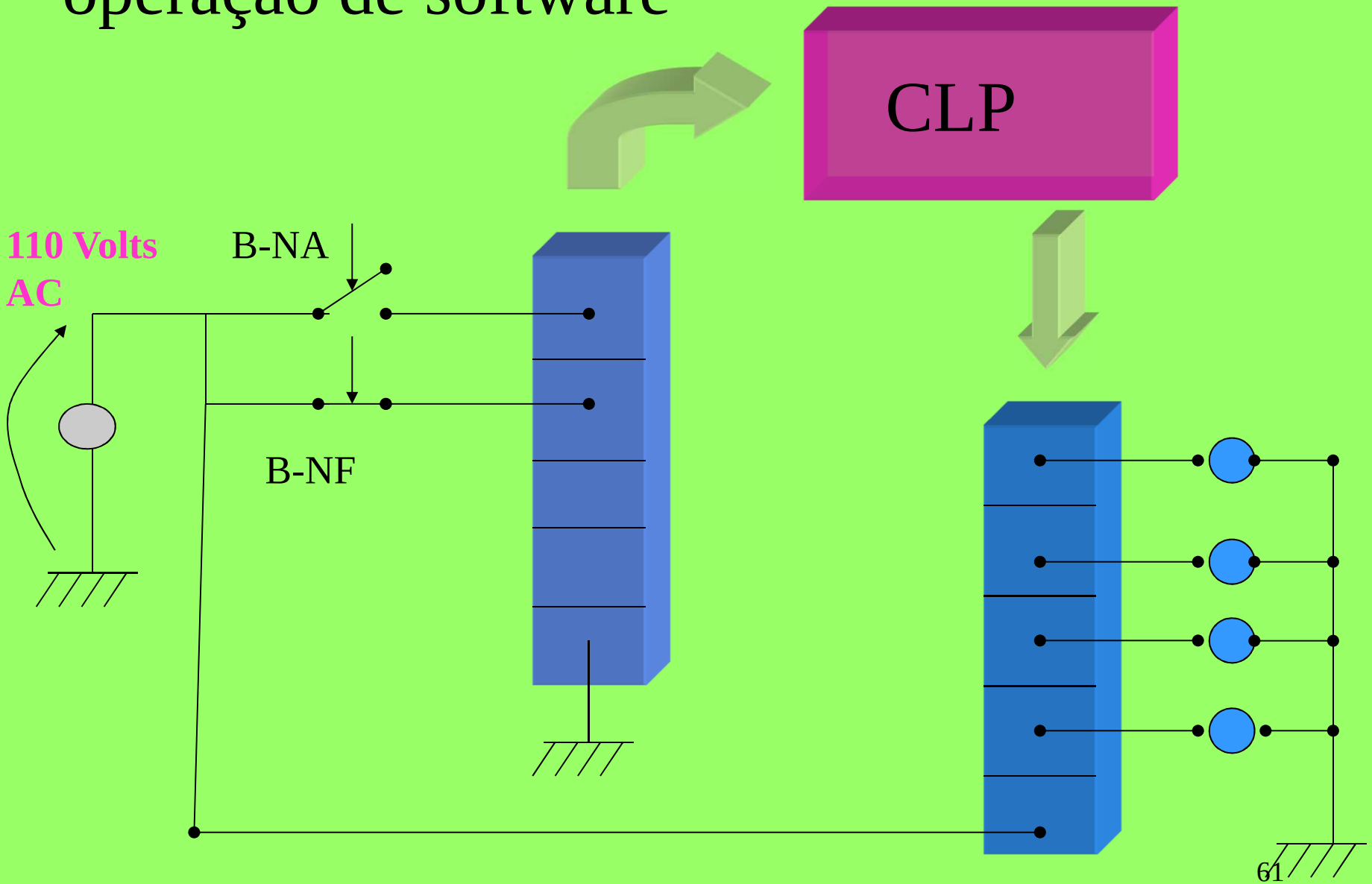


Circuito de Comando

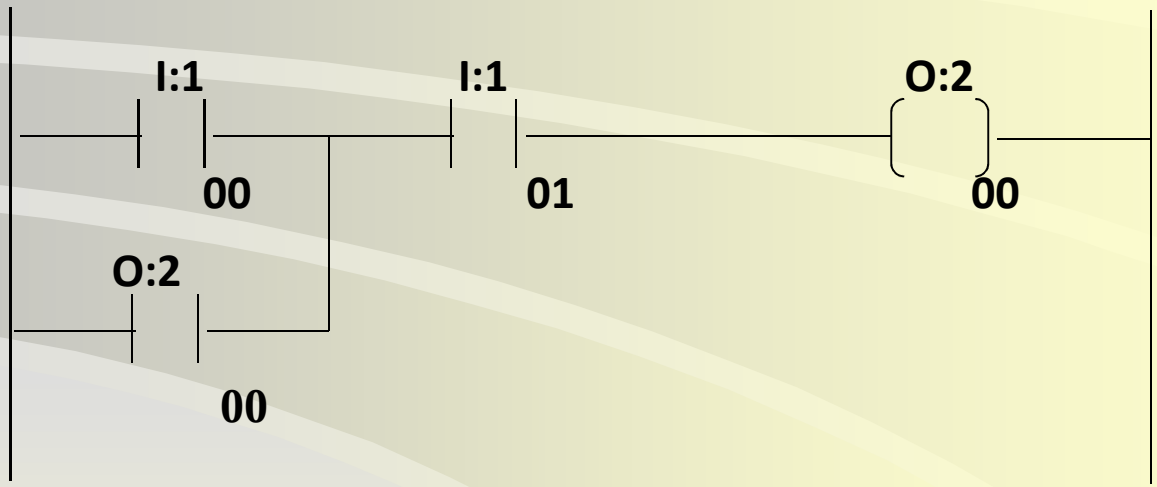
Circuito elétrico de comando



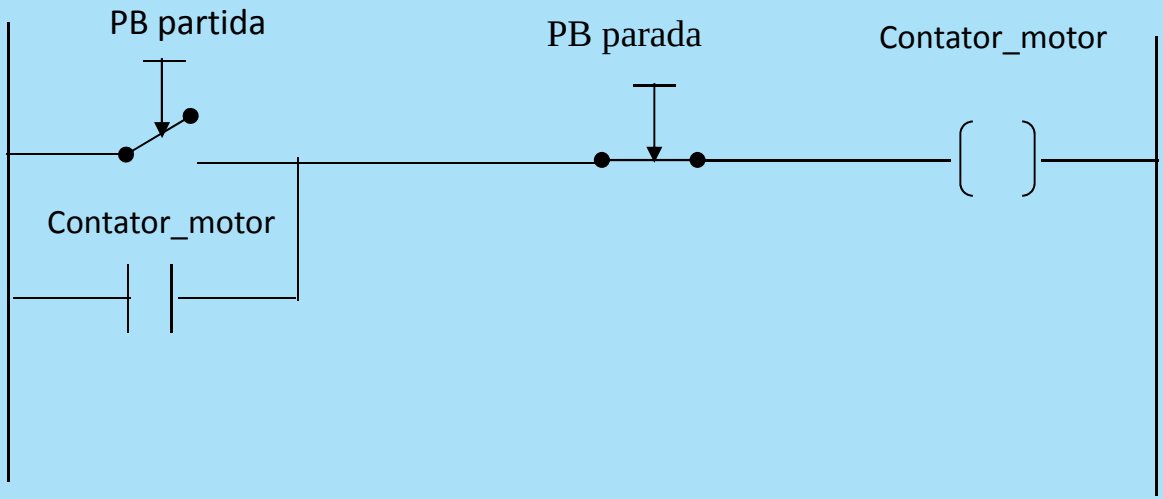
Exemplo de hardware e de operação de software



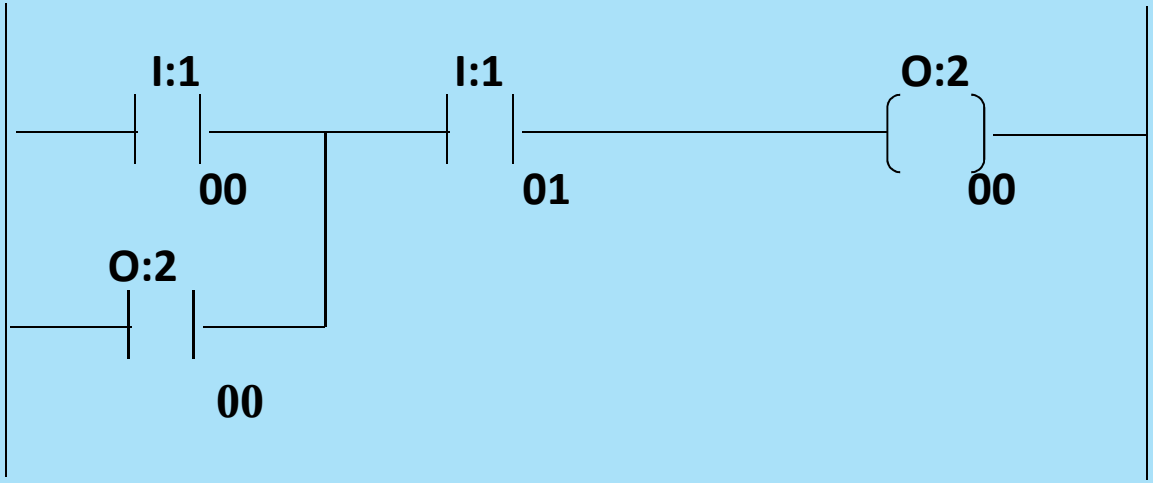
Programa em ladder



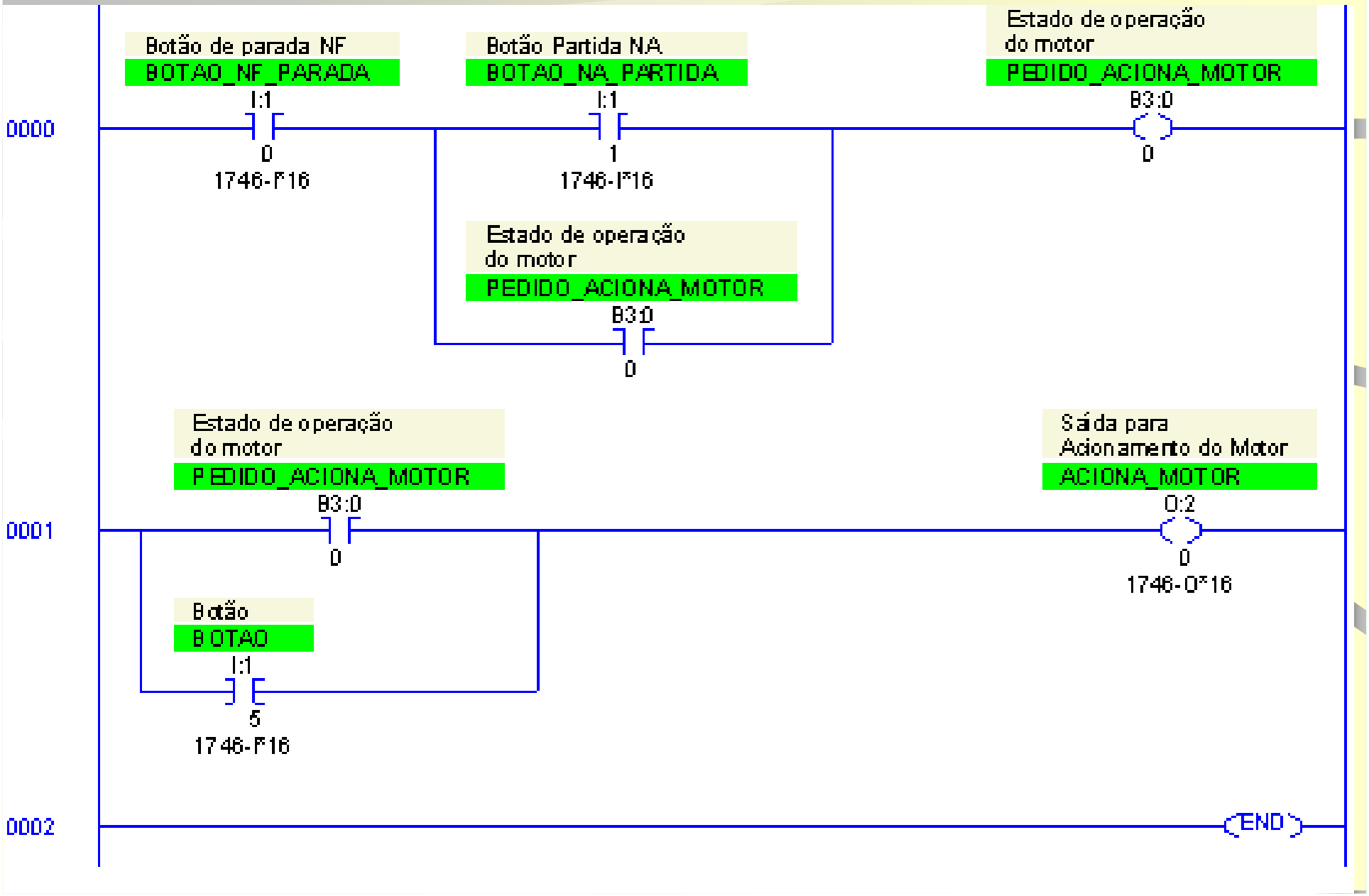
Circuito elétrico de comando



Programa em ladder

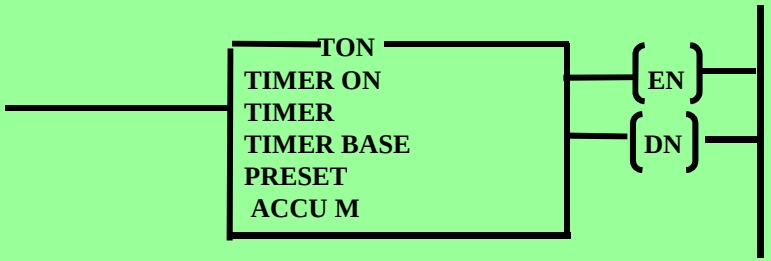


Exemplo da programa para Partida/Parada com Impulso/JOG



Instrução TON → Temporizador

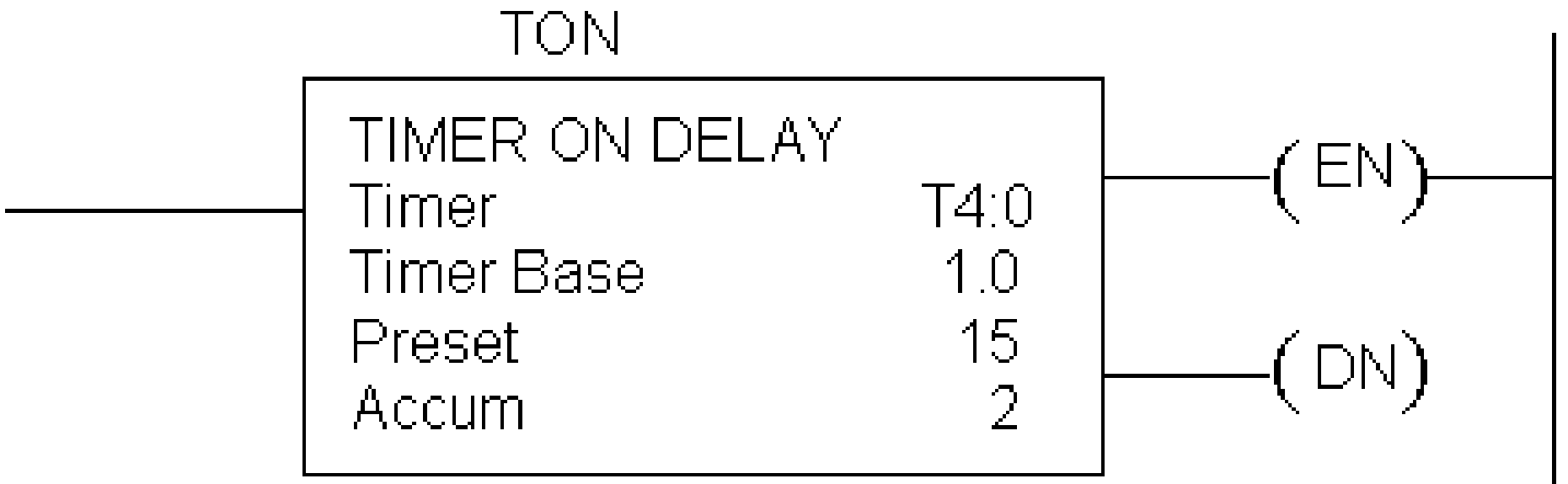
Inicia a contagem nos intervalos da base de tempo selecionada quando a condição da linha se torna verdadeira. O bit EN é colocado no estado lógico 1 cada vez que a instrução é acionada. O bit DN é colocado no estado lógico 1 quando o valor ACCUM for igual ao valor PRESET. A instrução Temporizador ocupa três palavras da memória.



Temporização

Instrução TON - Temporizador crescente sem retenção à Energização (Timer ON Delay)

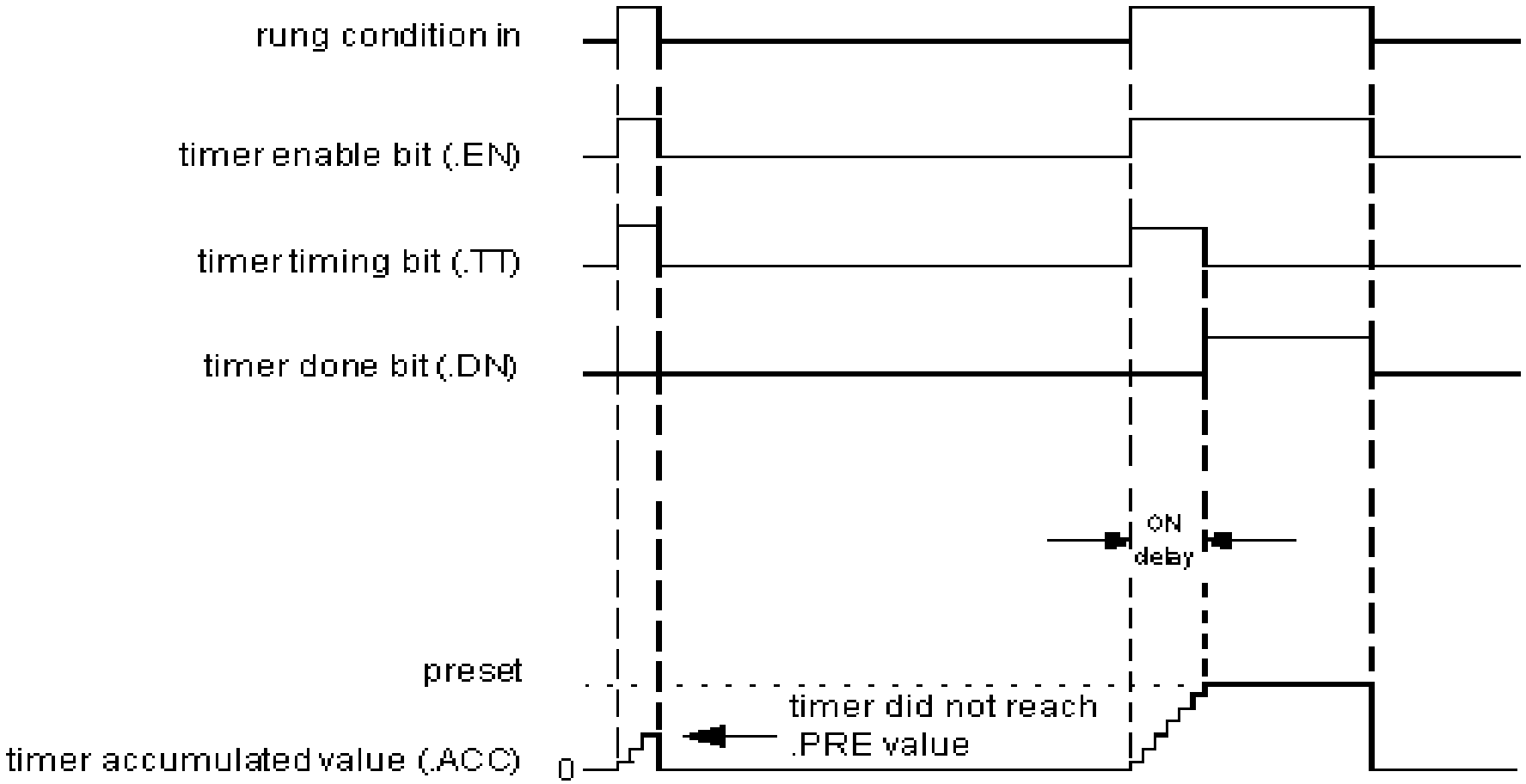
Caso *haja* continuidade lógica da linha, a instrução TON inicia uma contagem de tempo, baseada nos intervalos da base de tempo que deve ter sido selecionada durante a programação da instrução.



(ACCUM = PRESET) ⇒ (DN = 1)

Temporização

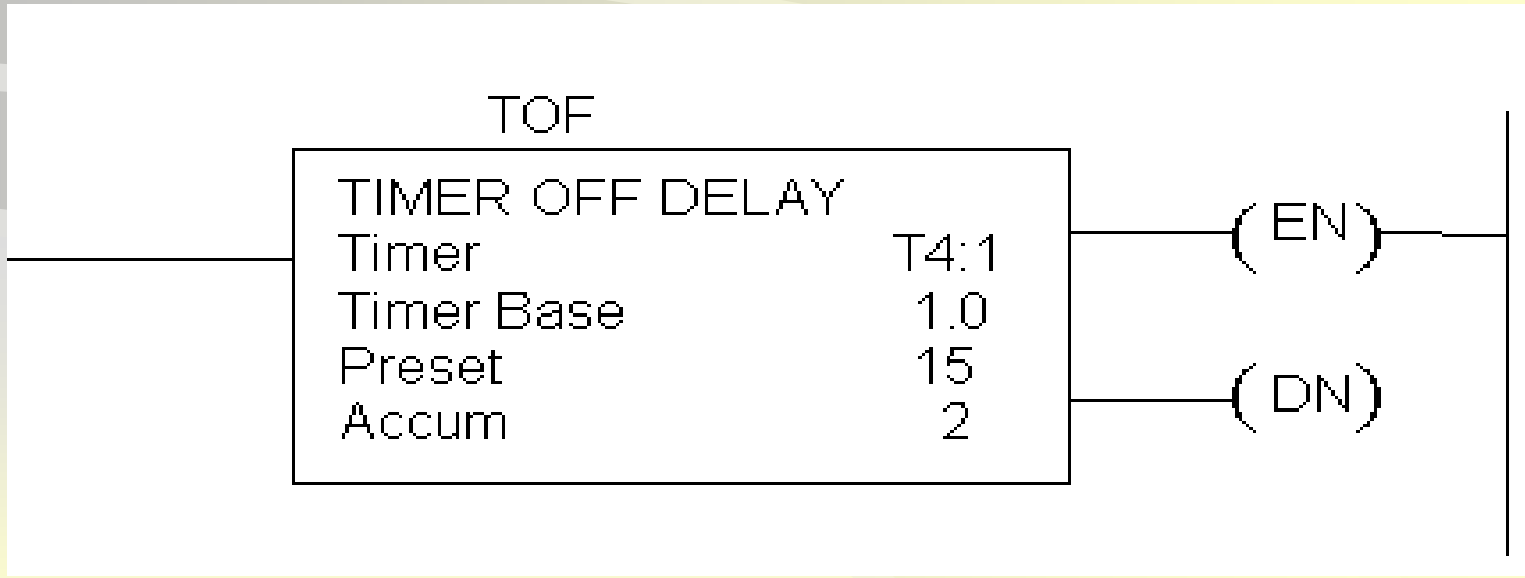
Instrução TON - Temporizador crescente sem retenção à Energização (Timer ON Delay)



Temporização

Instrução TOF - Temporizador crescente sem retenção à Desenergização (Timer OFf Delay)

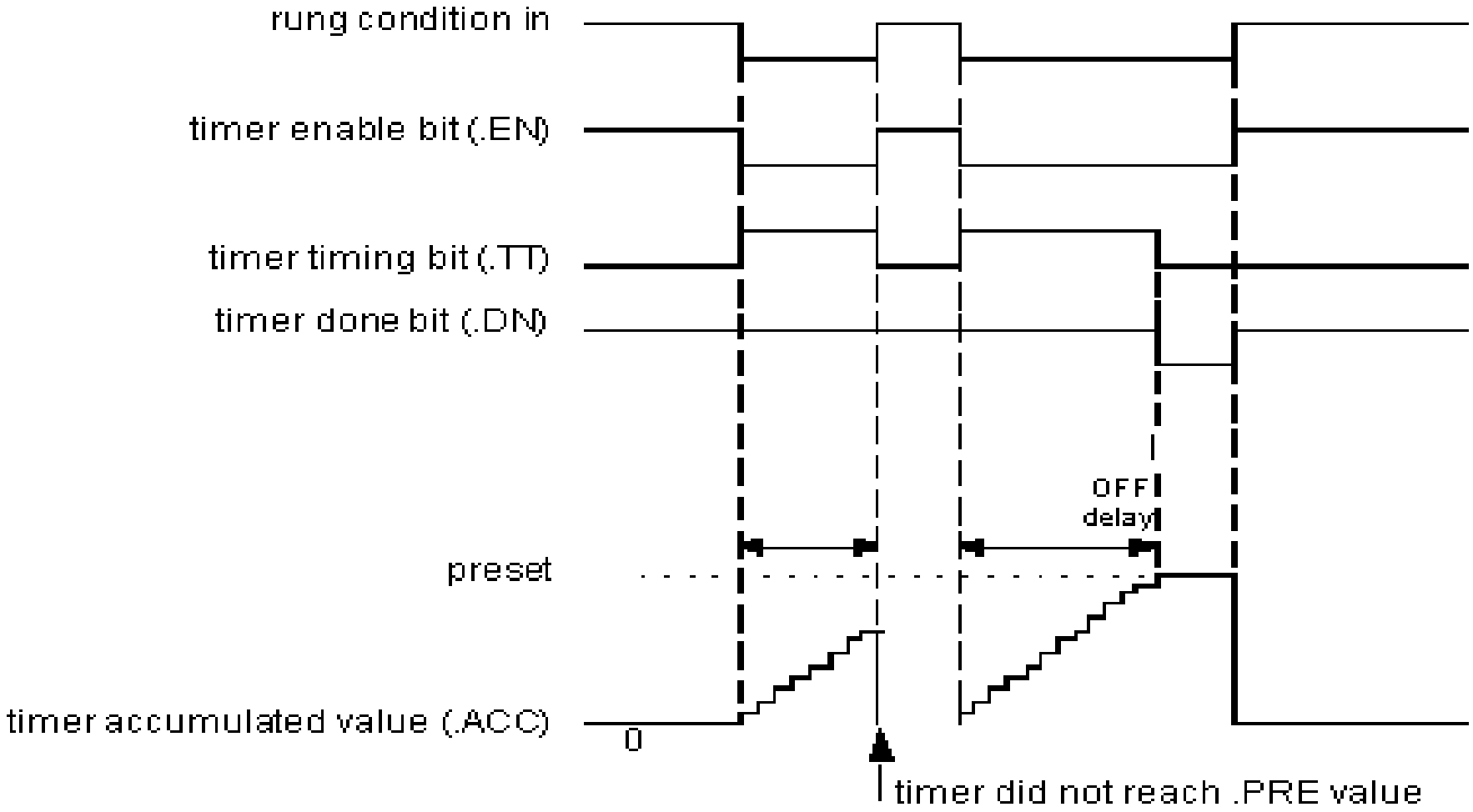
Caso *não haja* continuidade lógica da linha, a instrução TOF inicia uma contagem de tempo baseada nos intervalos de tempo da base de tempo selecionada durante a programação da instrução.



(ACCUM = PRESET) ⇒ (DN = 1)

Temporização

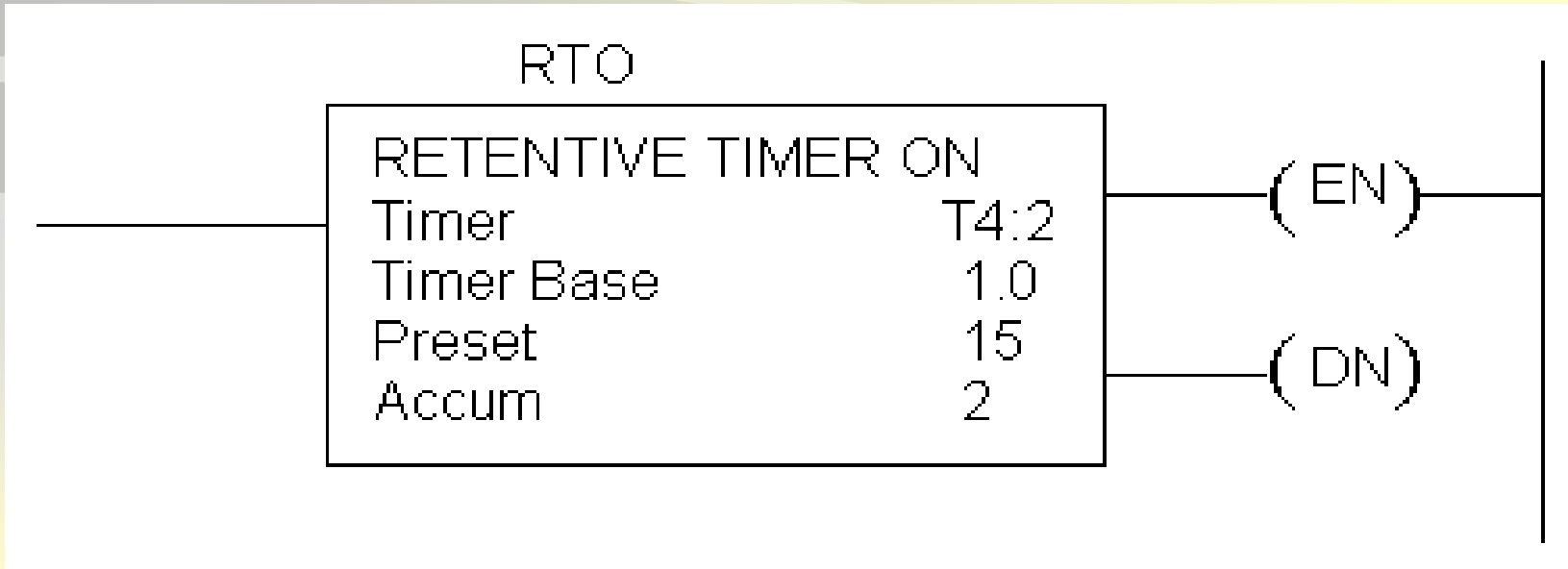
Instrução TOF - Temporizador crescente sem retenção à Desenergização (Timer OFf Delay)



Instrução RTO - Temporizador crescente com retenção

(Retentive Timer On)

O bit EN é colocado no estado lógico 1 cada vez que a instrução é acionada após instrução RTO. O bit DN é colocado no estado lógico 1 quando o “valor do registrador acumulado” (valor ACCUM) for igual ao “valor do registrador pré-selecionado” (valor PRESET).

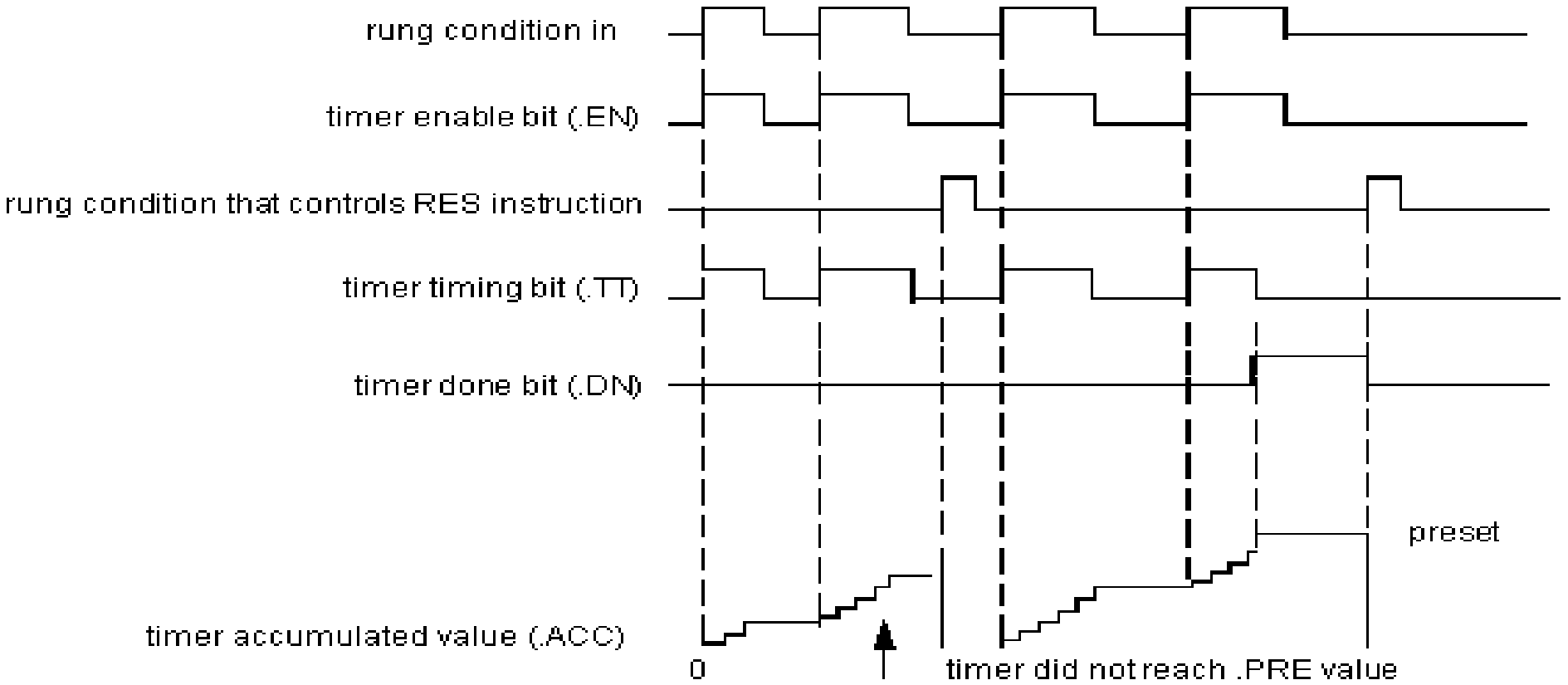


(ACCUM = PRESET) ⇒ (DN = 1)

Temporização

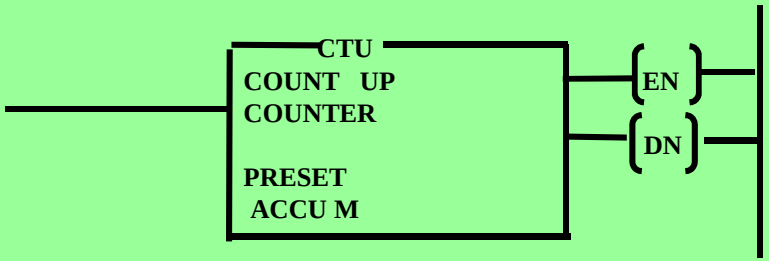
Instrução RTO - Temporizador crescente com retenção

(Retentive Timer On)



Instrução CTU → Contador crescente

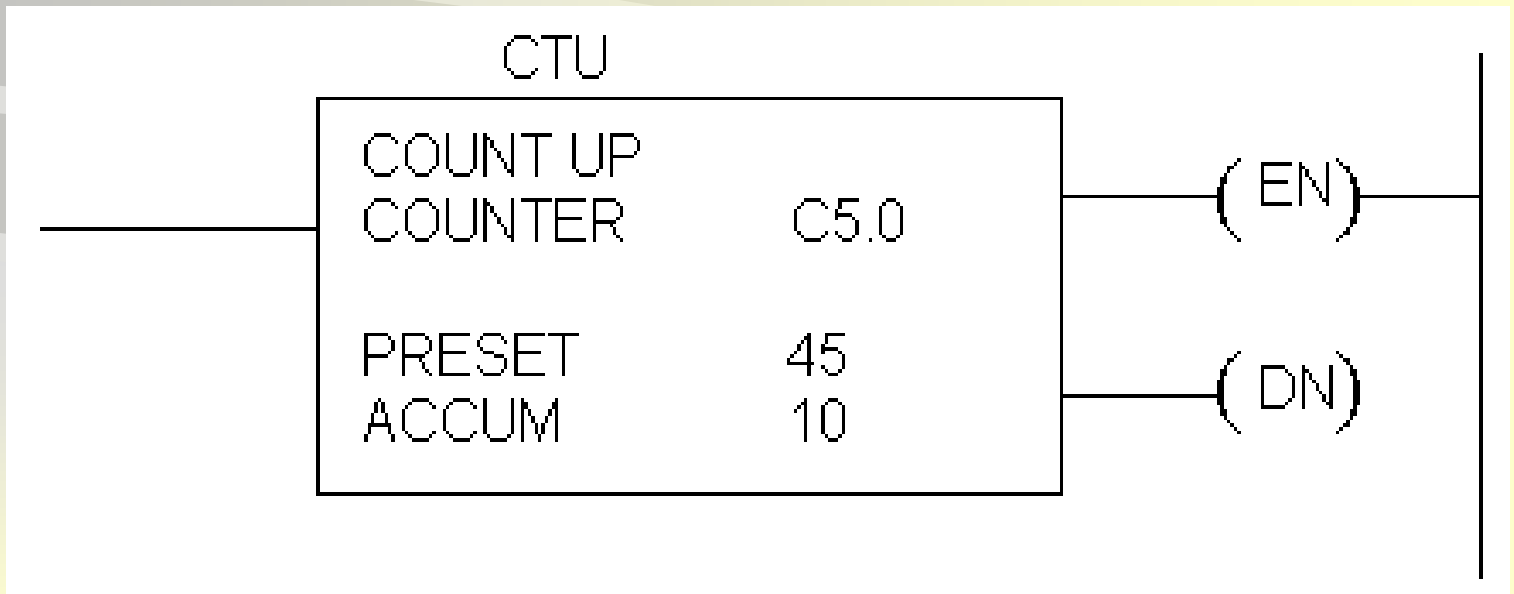
Incrementa o valor ACCUM a cada transição de falsa para verdadeira da condição lógica da linha em que a instrução está inserida.



Contagem

Instrução CTU - Contador crescente (CounT Up)

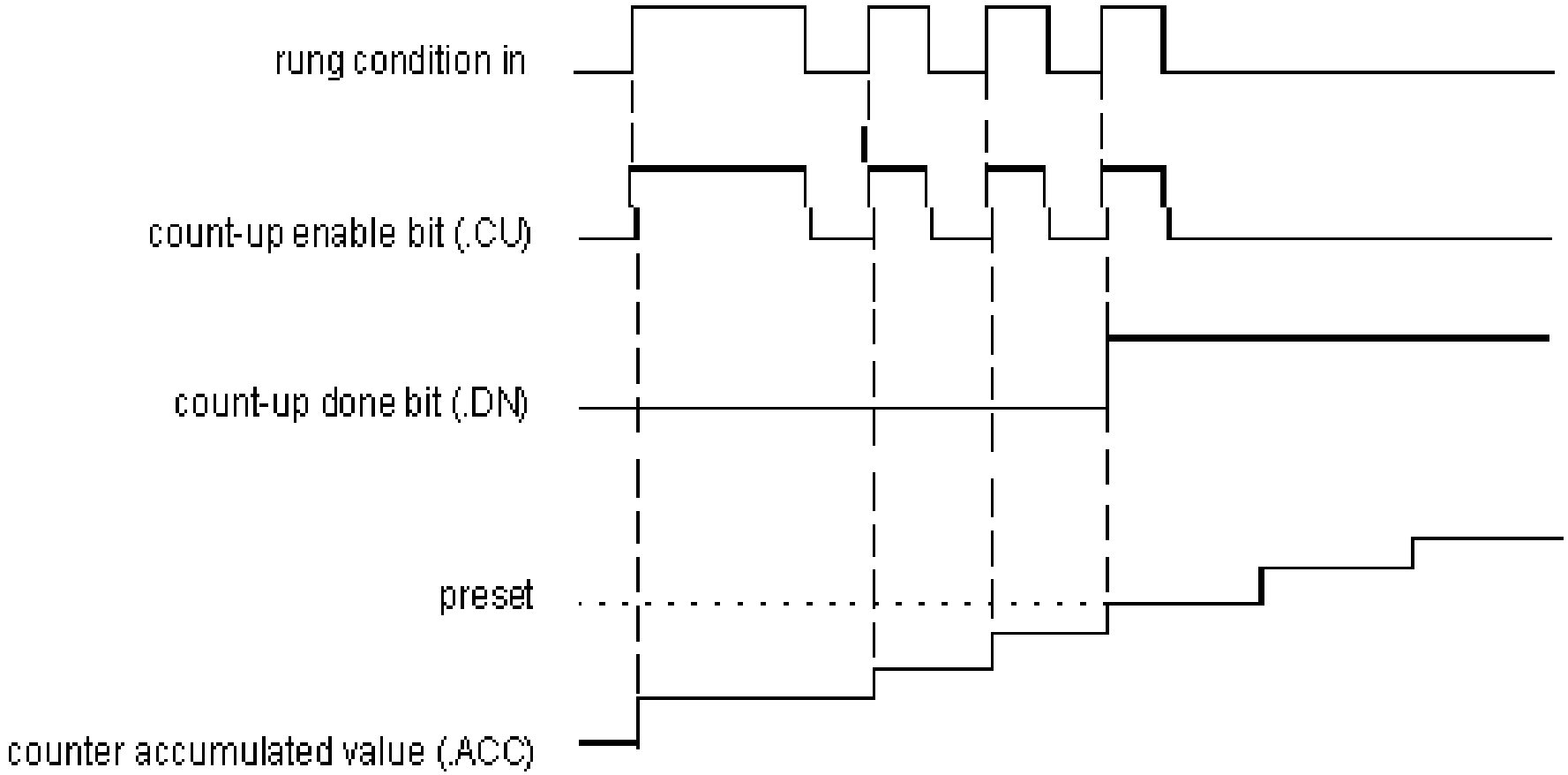
A cada transição da condição lógica da linha, de falsa para verdadeira, a instrução CTU incrementa o “valor do registrador acumulado” (valor ACCUM). Quando o valor ACCUM for igual ao “valor do registrador pré-selecionado” (valor PRESET), a instrução CTU coloca o bit DN no valor lógico 1.



(ACCUM = PRESET) ⇒ (DN = 1)

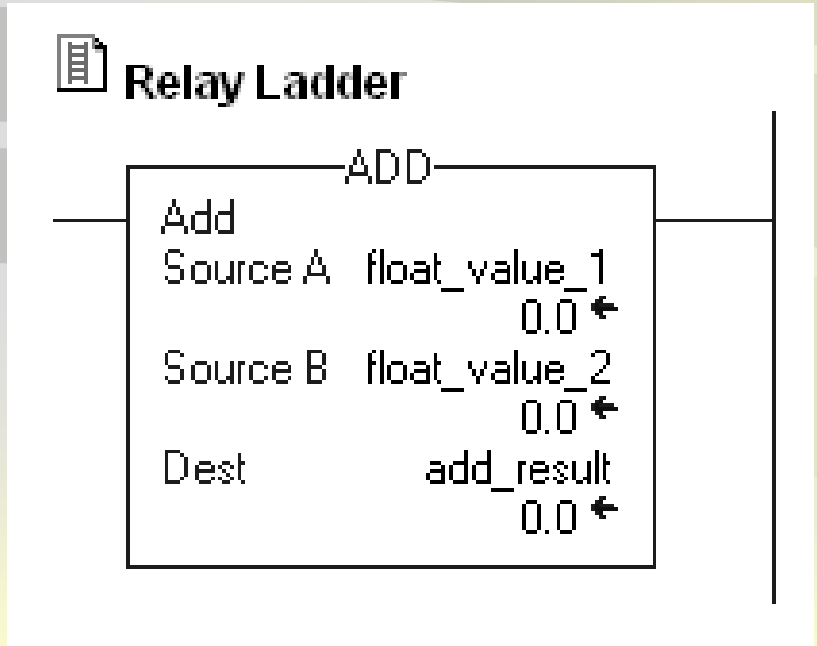
Contagem

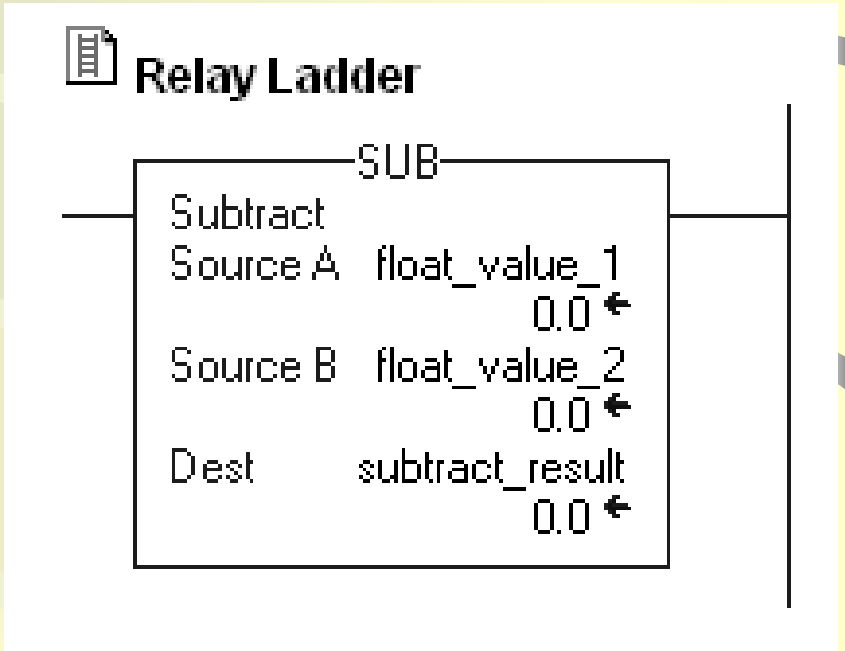
Instrução CTU - Contador crescente (CounT Up)



Aritméticas

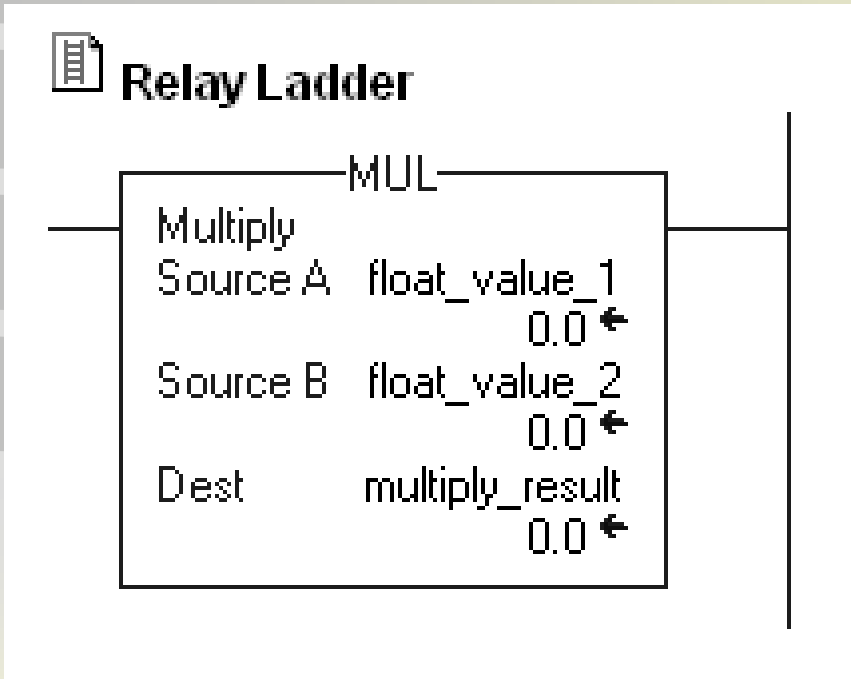
Instrução ADD - Adição e Instrução SUB - Subtração

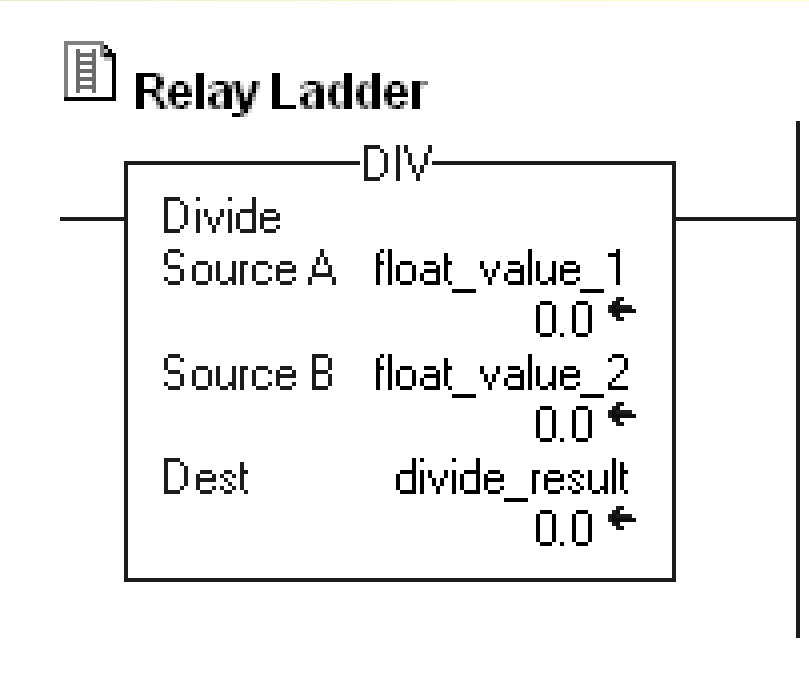




Aritméticas

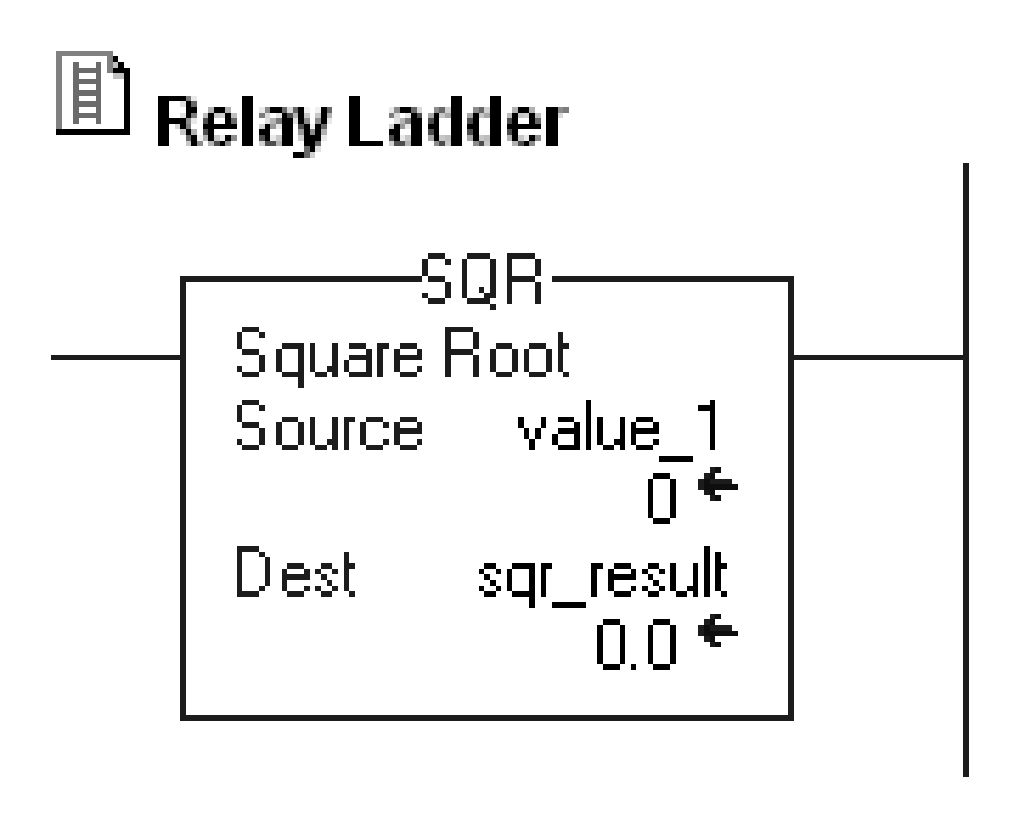
Instrução MUL - Multiplicação e Instrução DIV - Divisão





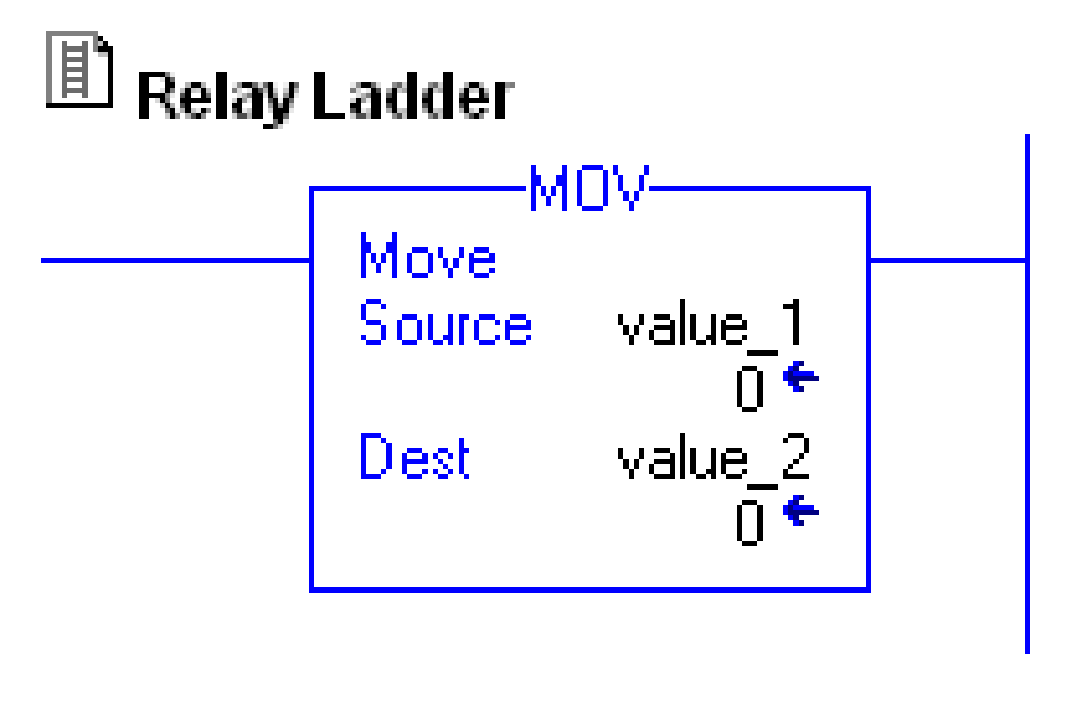
Aritméticas

Instrução SQR - Raiz Quadrada (SQuare Root)



Manipulação de dados

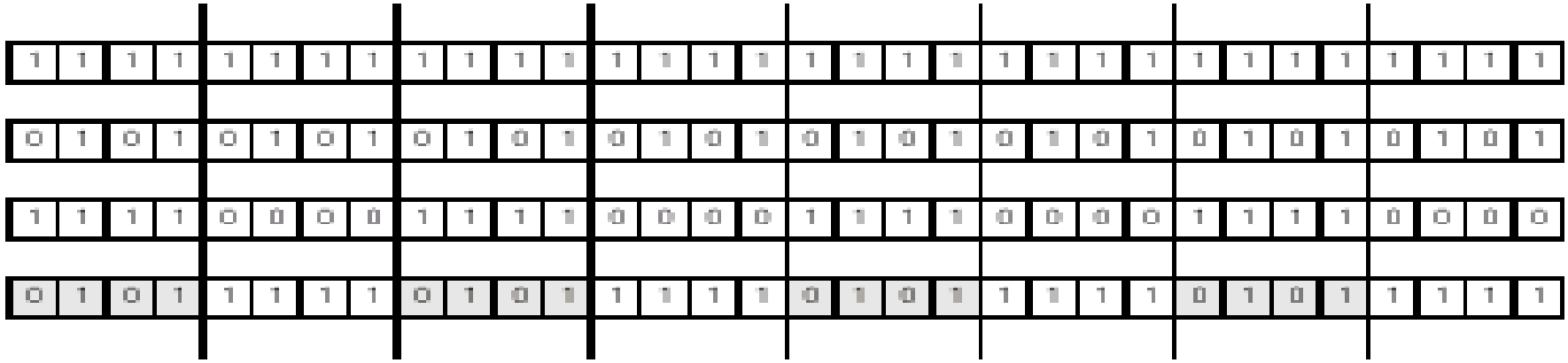
Instrução MOV - Mover (MOVe)



Manipulação de dados

Instrução MVM - Mover com máscara (MoVe with Mask)

Relay Ladder



The shaded boxes show the bits that changed in value_b.

MVM

Masked Move

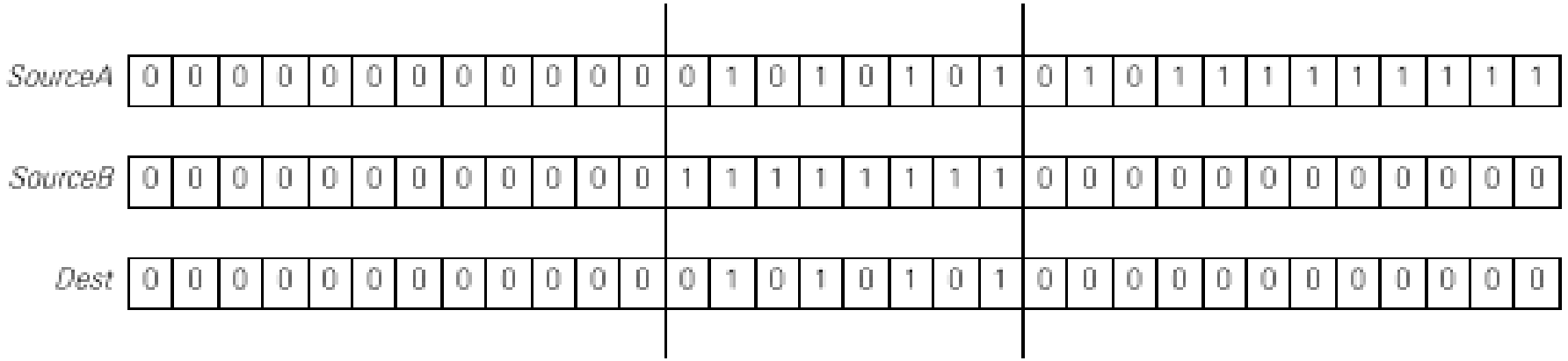
Source — value_a
2#0101_0101_0101_0101_0101_0101_0101_0101

Mask — mask_2
2#1111_0000_1111_0000_1111_0000_1111_0000

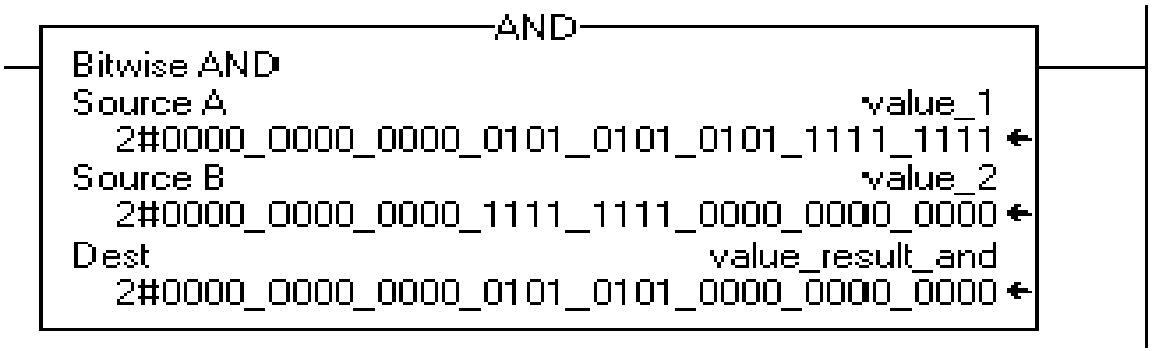
Dest — value_b
2#1111_1111_1111_1111_1111_1111_1111_1111

Manipulação de dados

Instrução AND - Lógica E (AND)

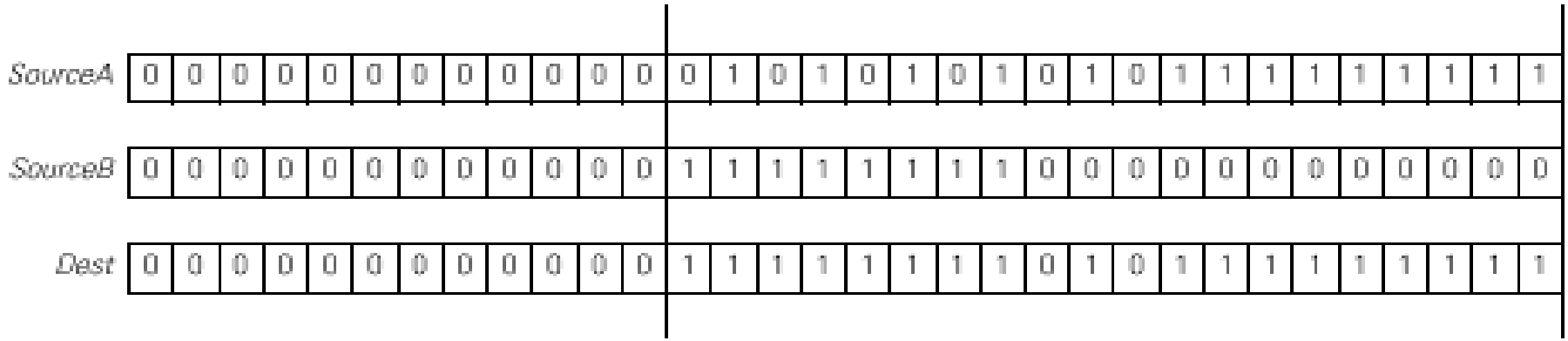


Relay Ladder

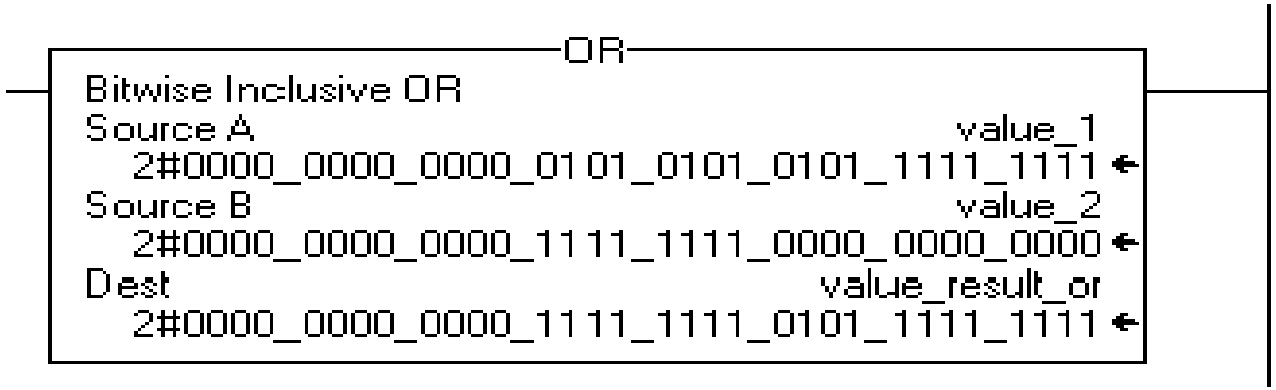


Manipulação de dados

) Instrução OR - Lógica OU (OR)

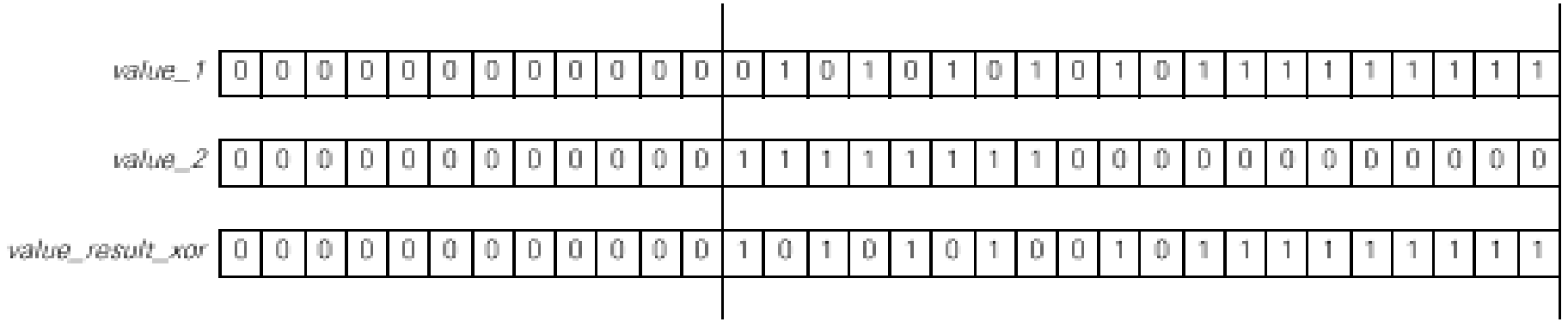


Relay Ladder

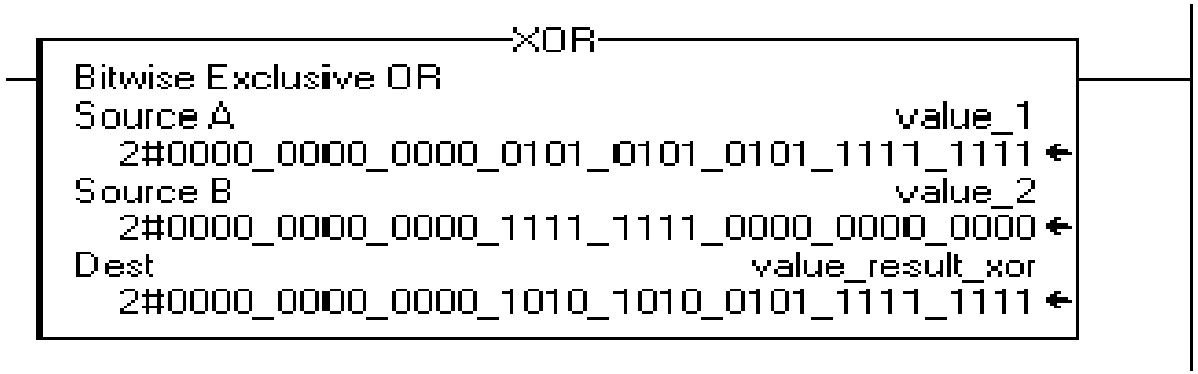


Manipulação de dados

Instrução XOR - Ou Exclusivo (eXclusive OR)



Relay Ladder

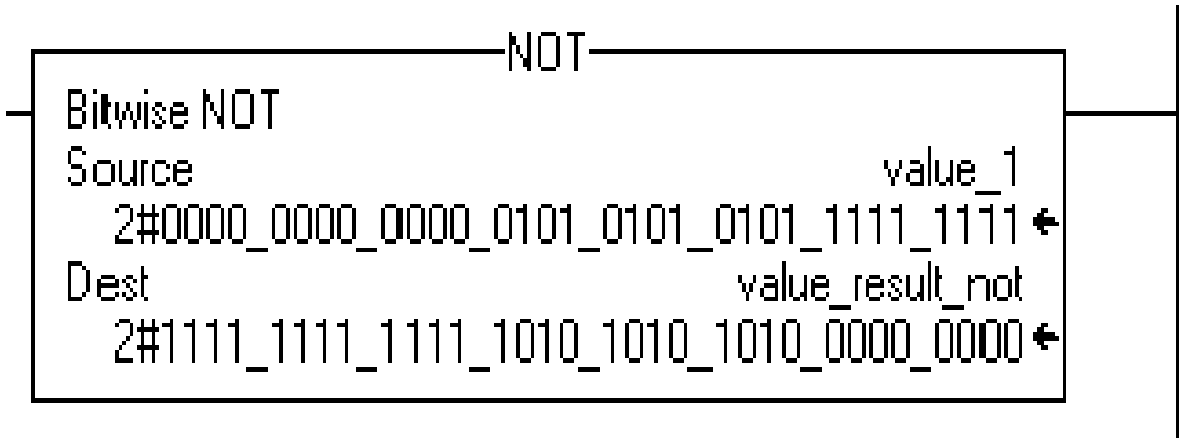


Manipulação de dados

Instrução NOT - Não lógico (NOT)

value_1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1
value_result_not	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0

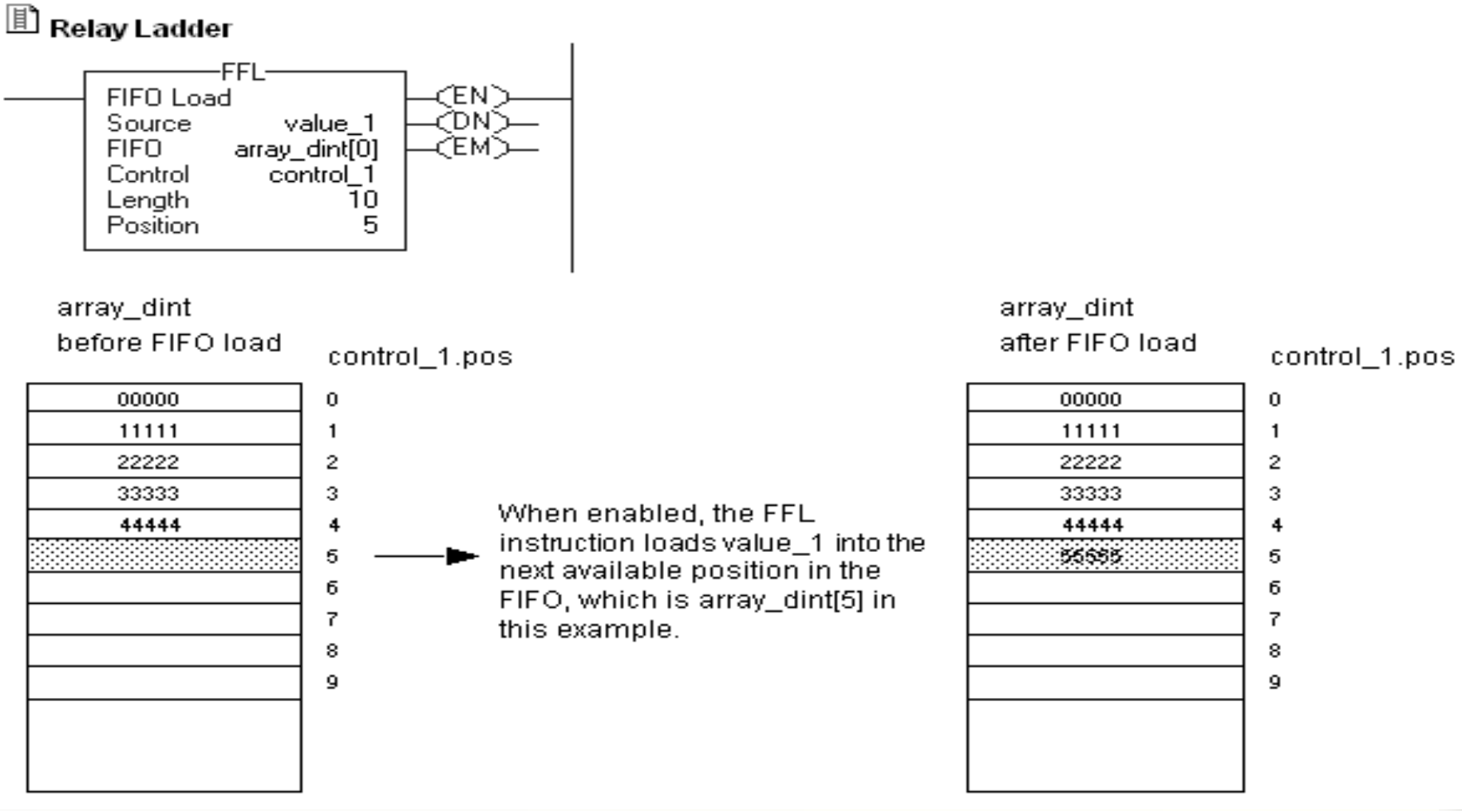
Relay Ladder



Manipulação de dados

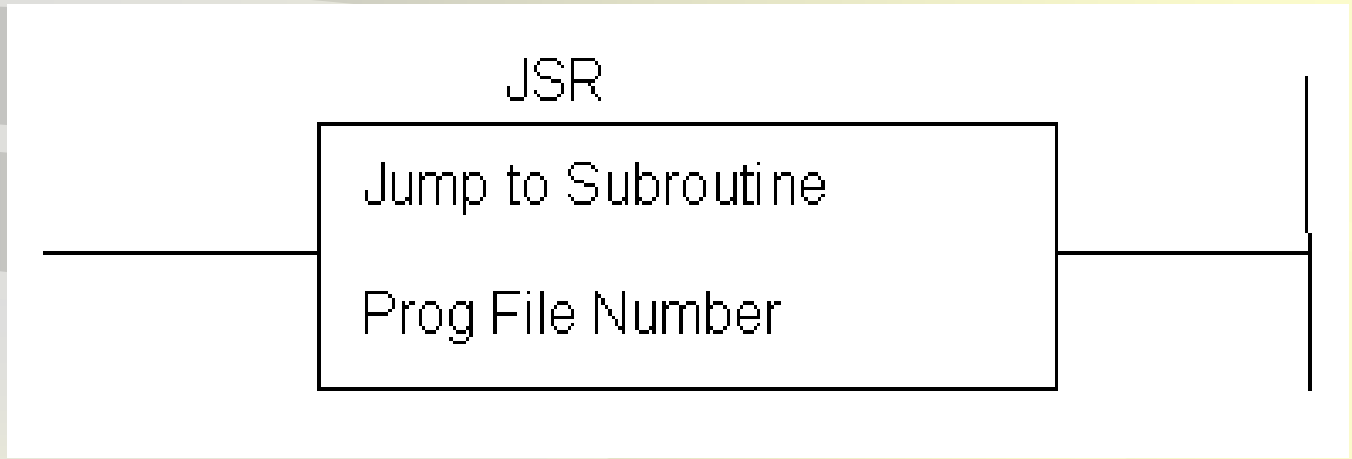
Instrução FFL - Primeiro a entrar primeiro a sair

(First in First out Load)



Controle de execução de programa

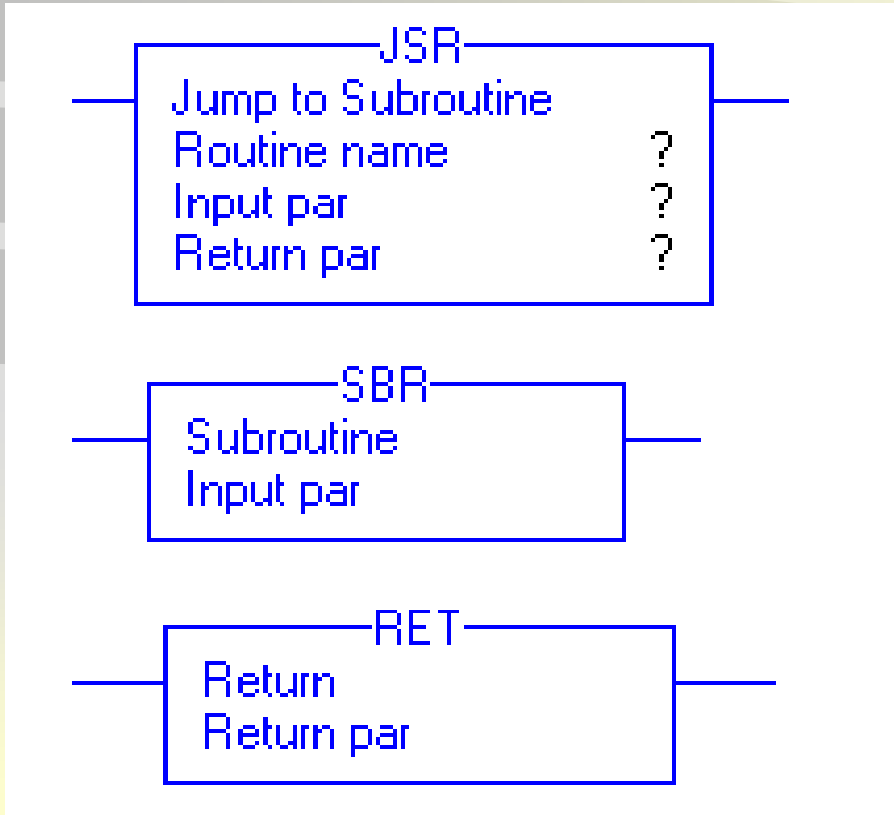
Instrução JSR - Pule para sub rotina (Jump to SubRoutine)

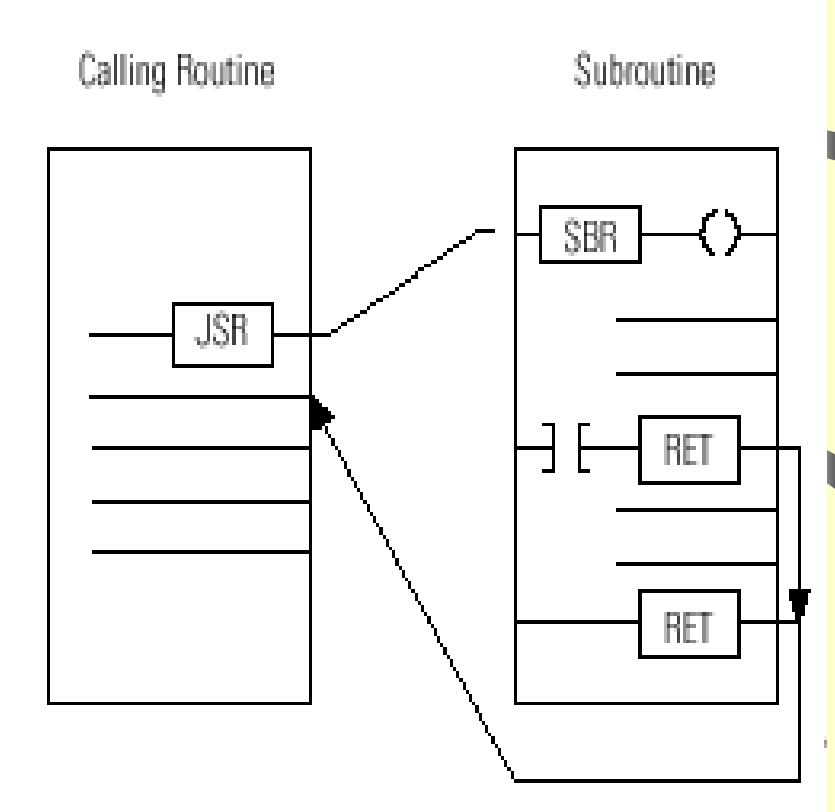


Controle de execução de programa

Instrução JSR - Pule para sub rotina (Jump to SubRoutine)

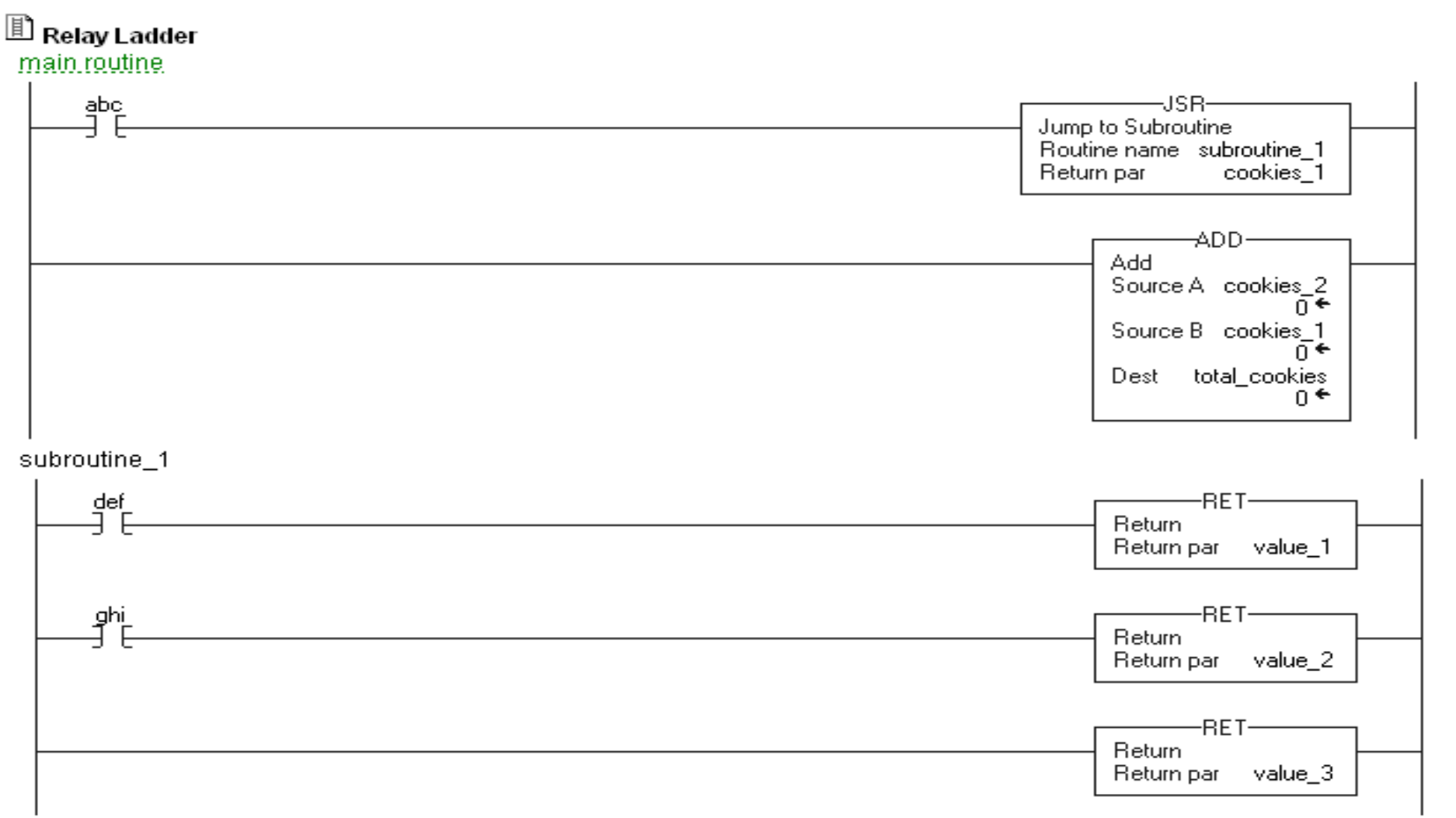
Instrução SBR - SuBRoutine e Instrução RET - RETurn)





Controle de execução de programa

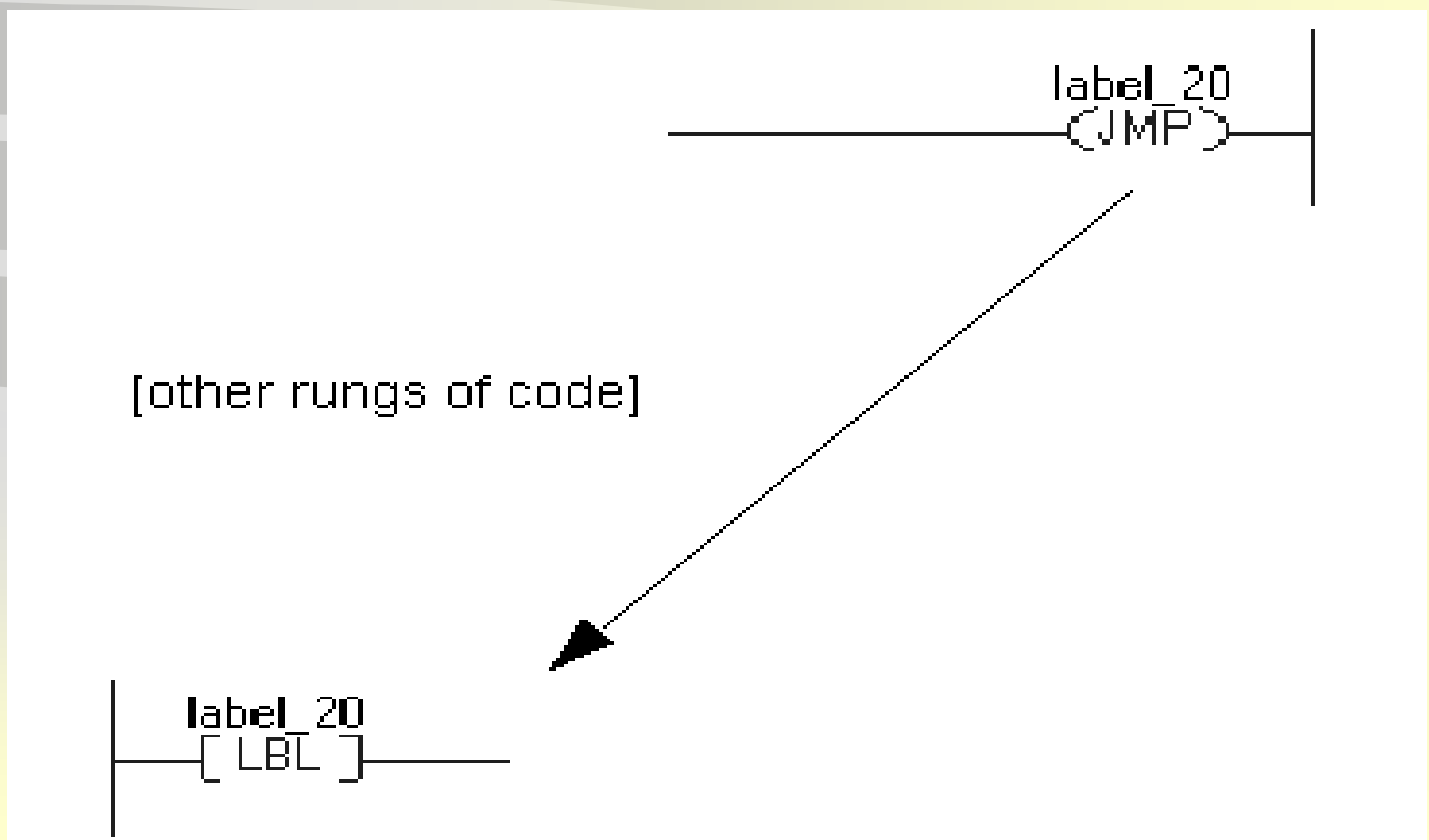
Instrução JSR - Pule para sub rotina (Jump to SubRoutine)



Controle de execução de programa

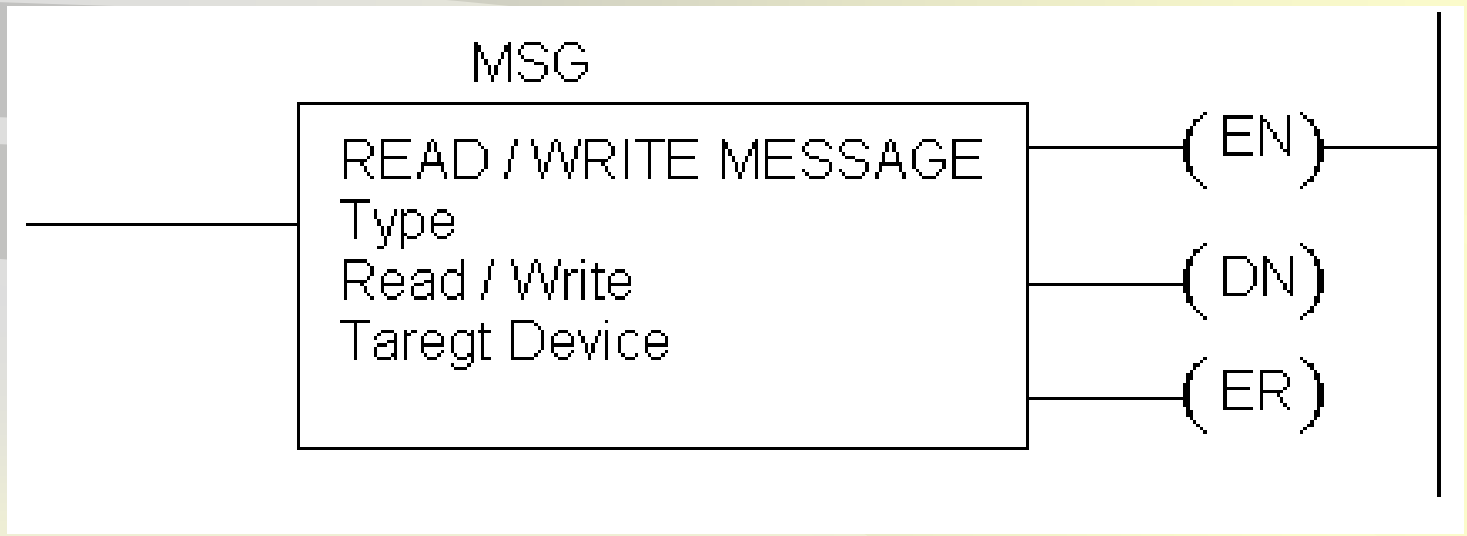
Instrução JMP - FOR (De) (JuMP) e

Instrução LBL - NEXT (Para) (LaBeL)



Transferência de dados

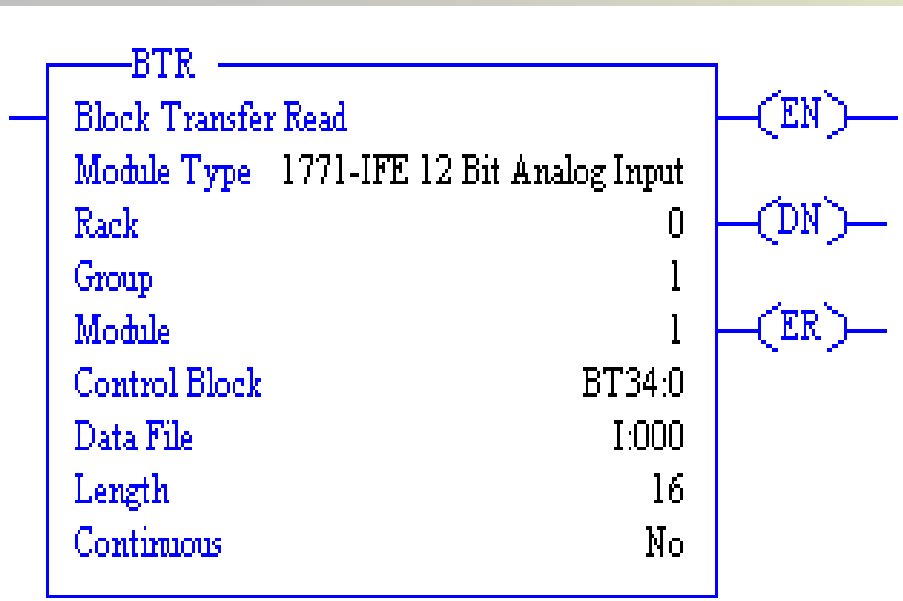
Instrução MSG - Mensagem (MeSsaGe)

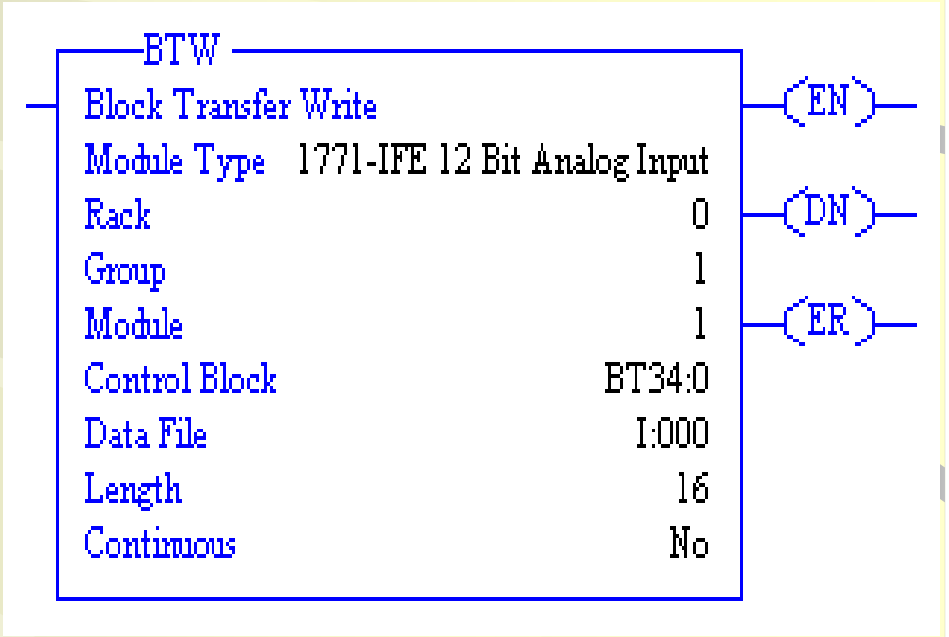


Transferência de dados

Instrução BTR - Leitura de Dados (Block Transfer Read)

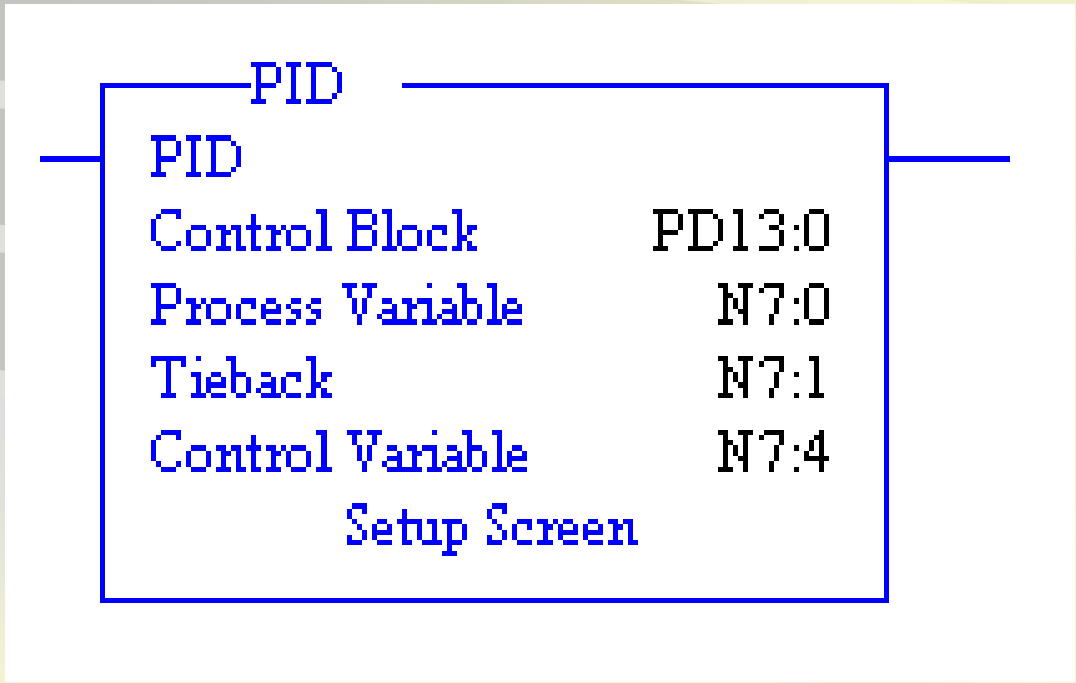
Instrução BTW - Escrita de Dados (Block Transfer Write)



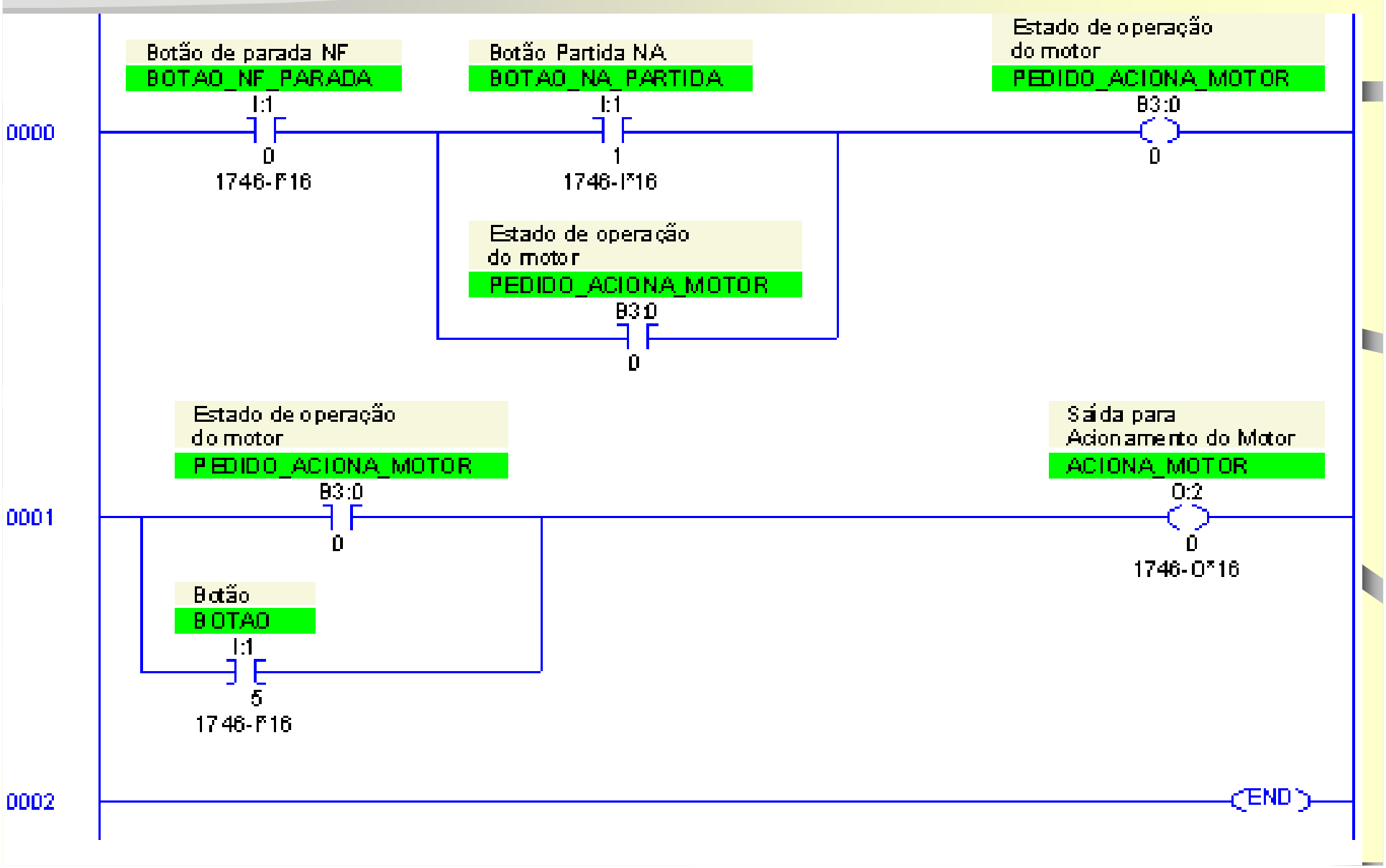


Avançadas

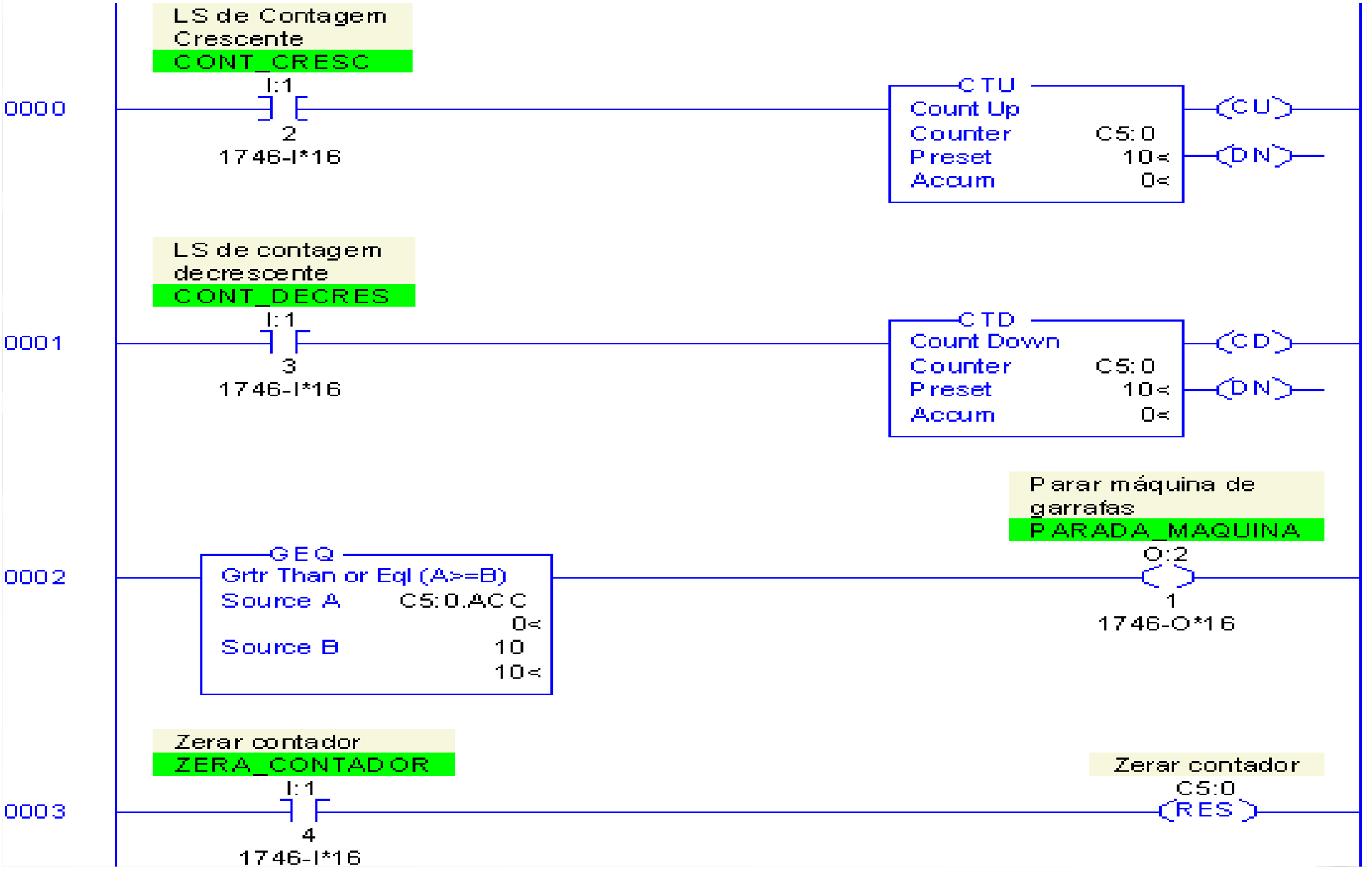
Instrução PID - Controle PID (Proporcional Integral Derivative)



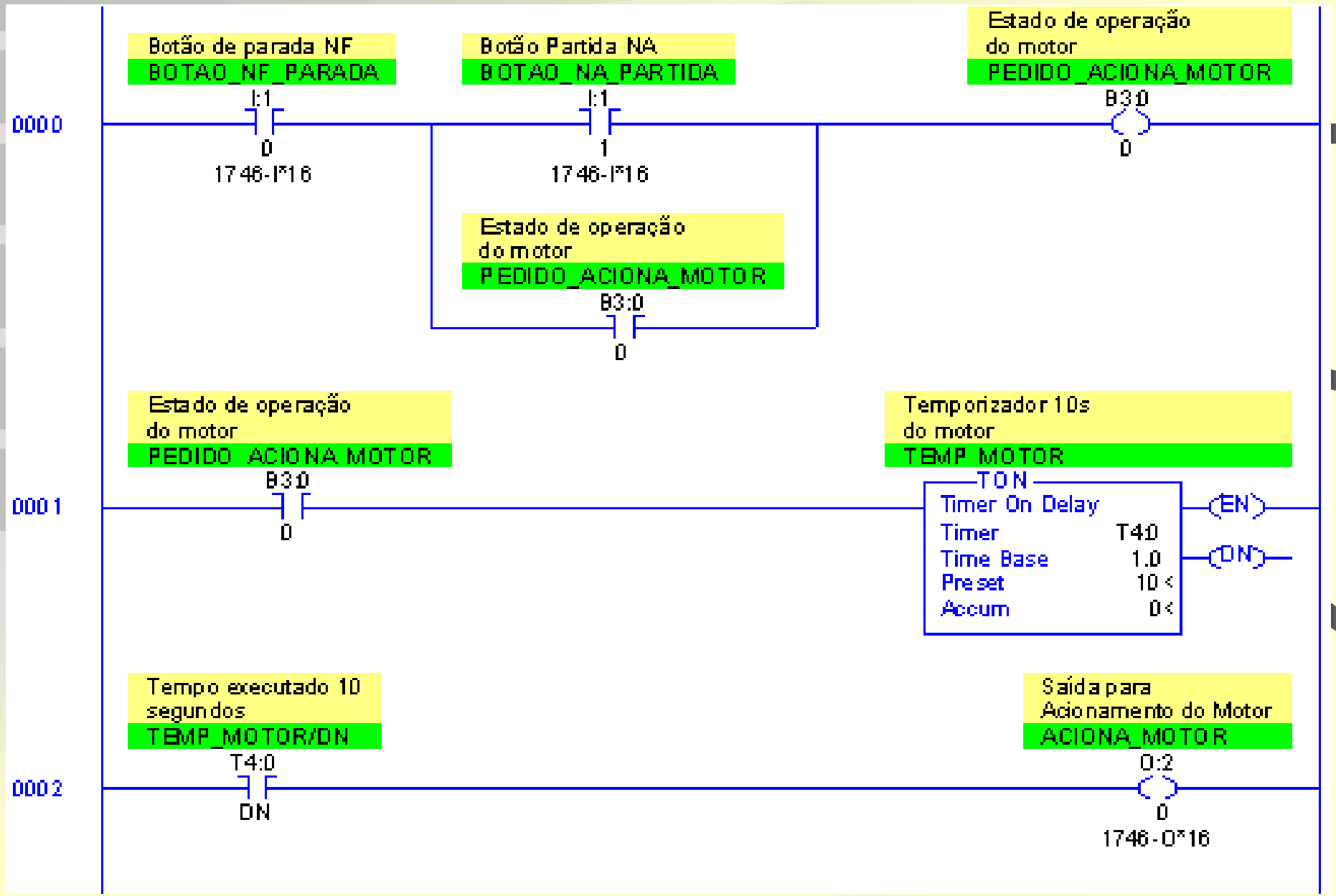
Partida/Parada com Impulso/JOG



Contagem Crescente/Decrescente

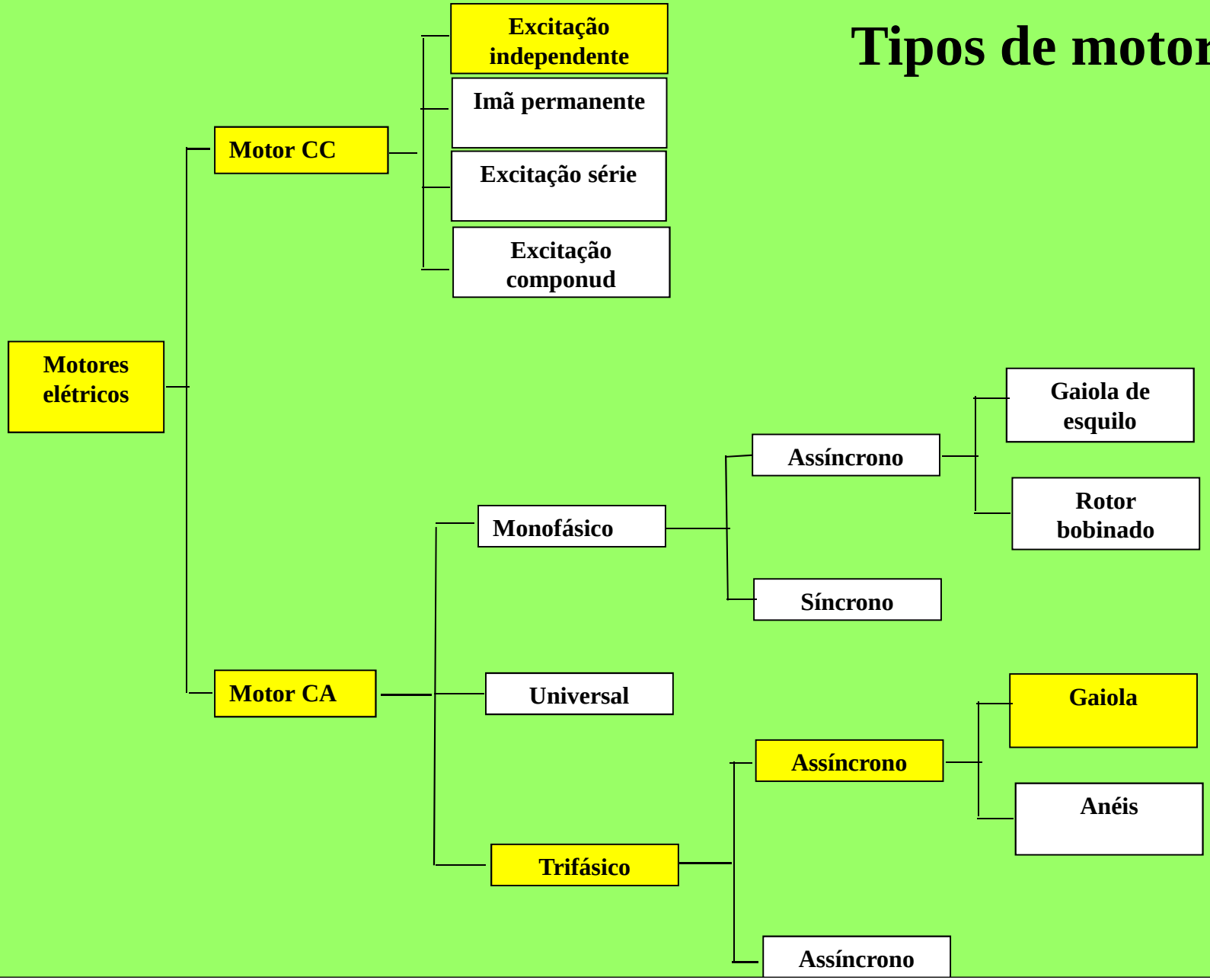


Temporizador na Energização

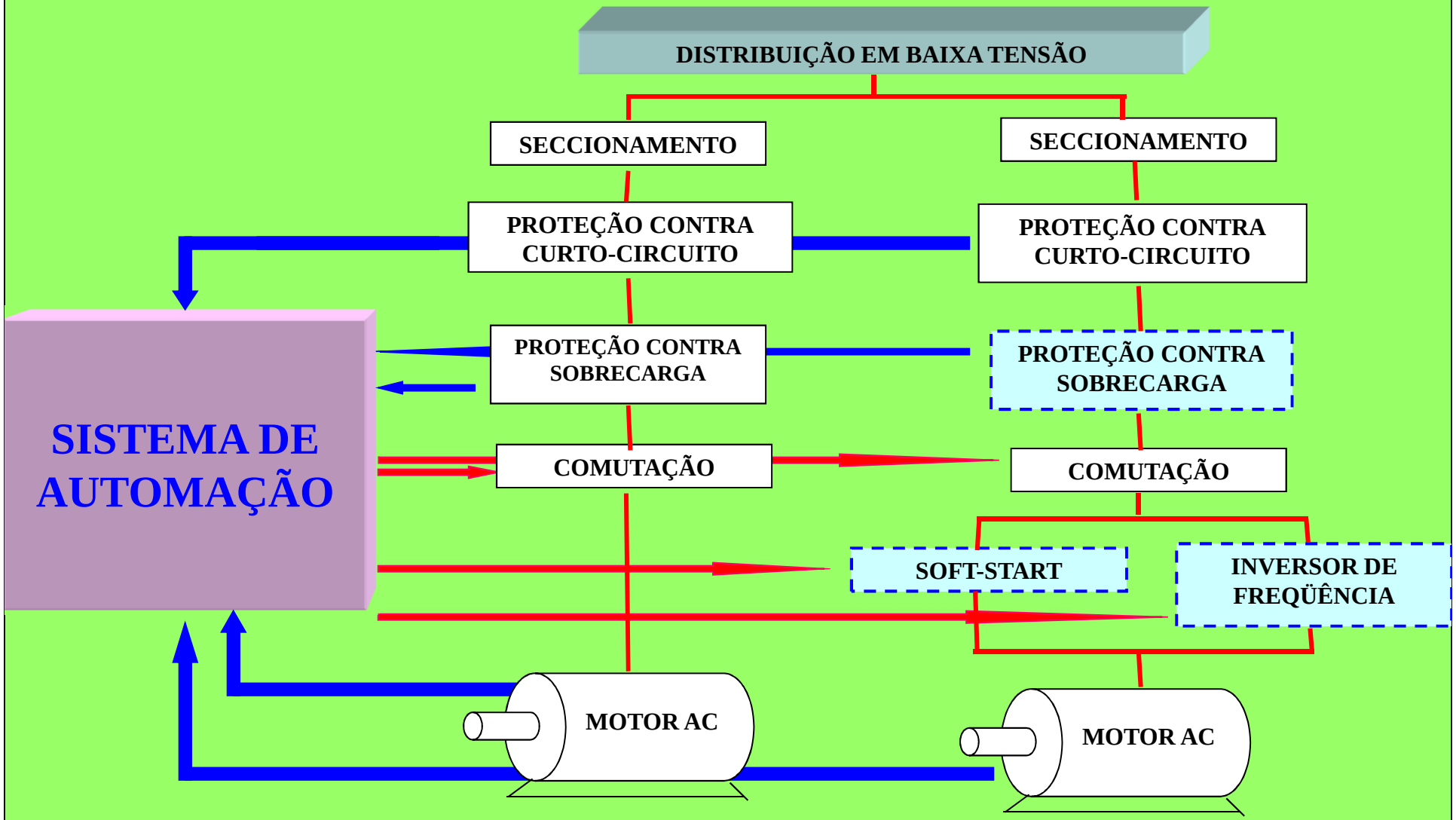


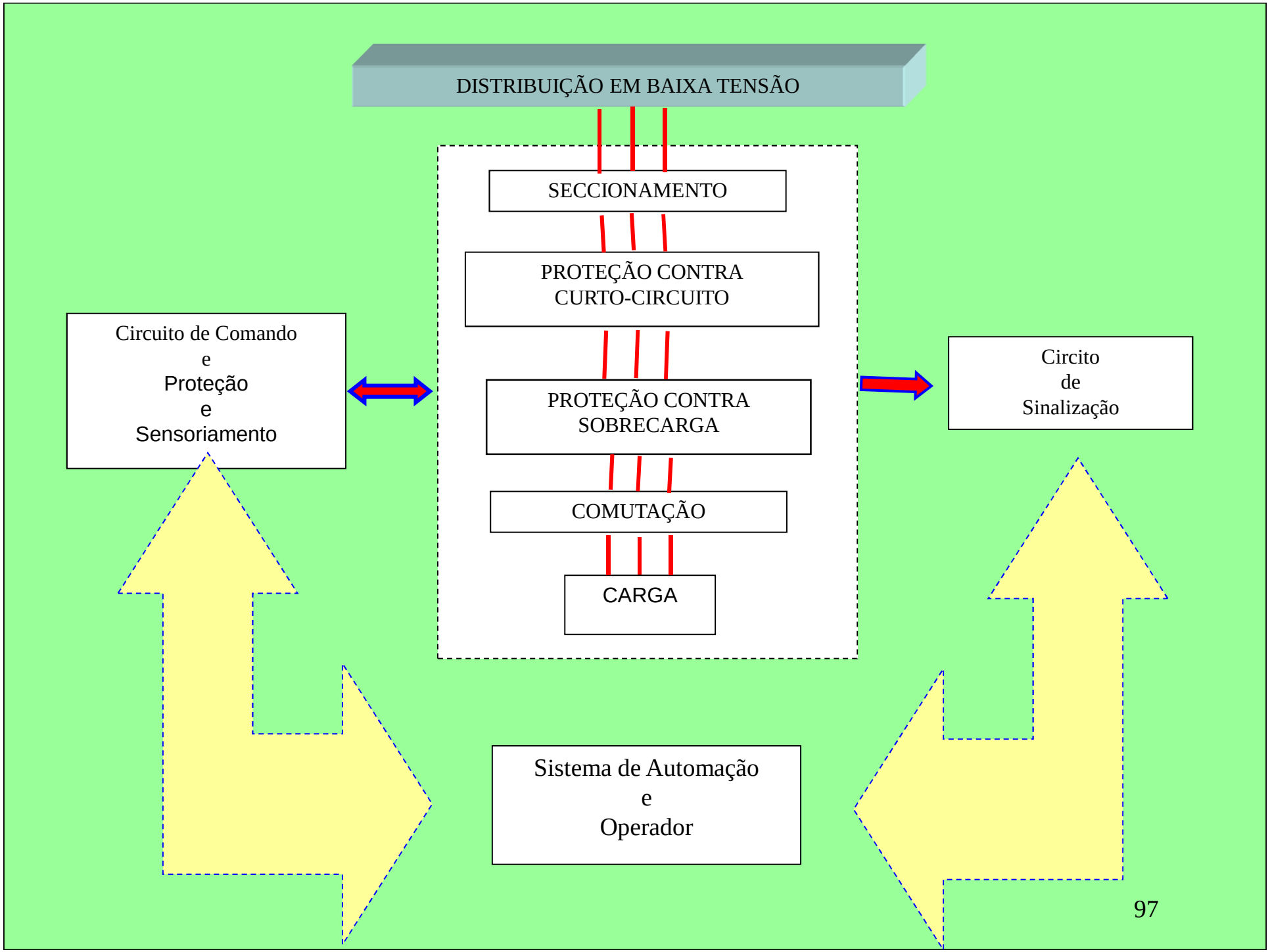
ACIONAMENTO DE MOTORES ELÉTRICOS

Tipos de motores



COMPONENTES DE UM PROJETO DE ACIONAMENTO



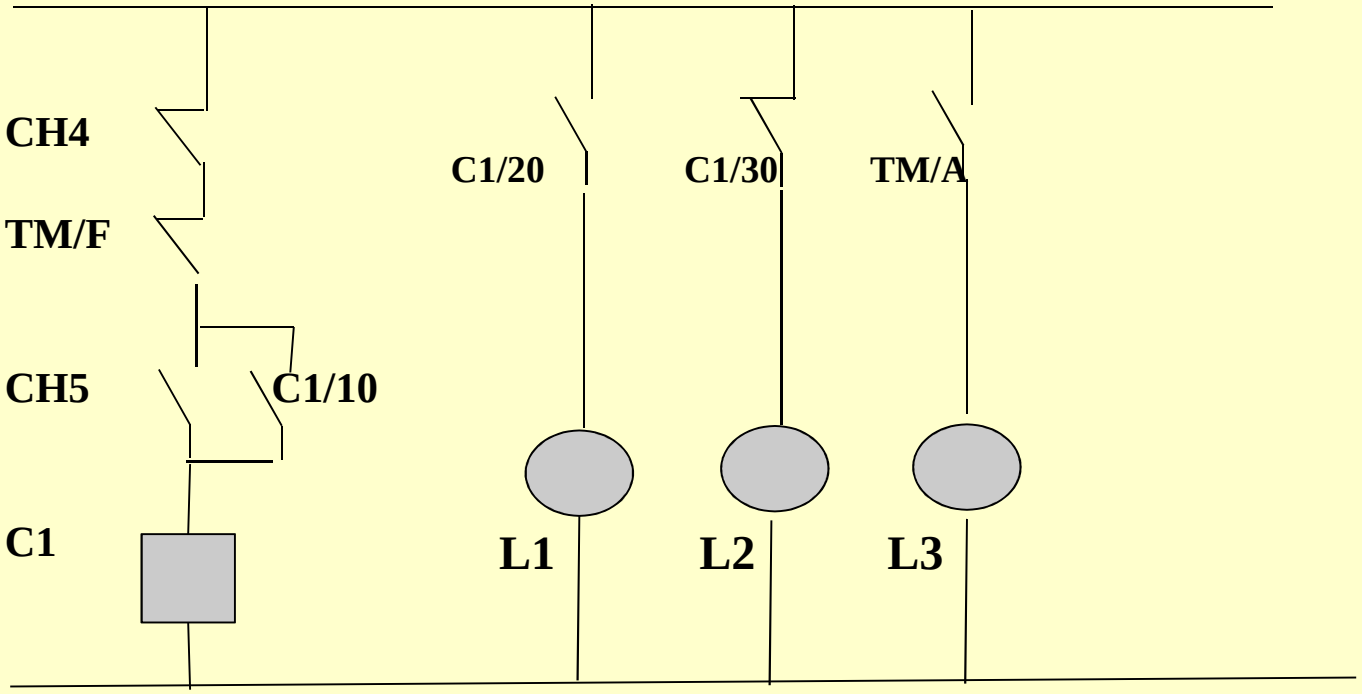

DISTRIBUIÇÃO EM BAIXA TENSÃO
SECCIONAMENTO
PROTEÇÃO CONTRA CURTO-CIRCUITO
PROTEÇÃO CONTRA SOBRECARGA
COMUTAÇÃO
CARGA
Circuito de Comando e Proteção e Sensoriamento
Circito de Sinalização
Sistema de Automação e Operador

EXERCÍCIOS:

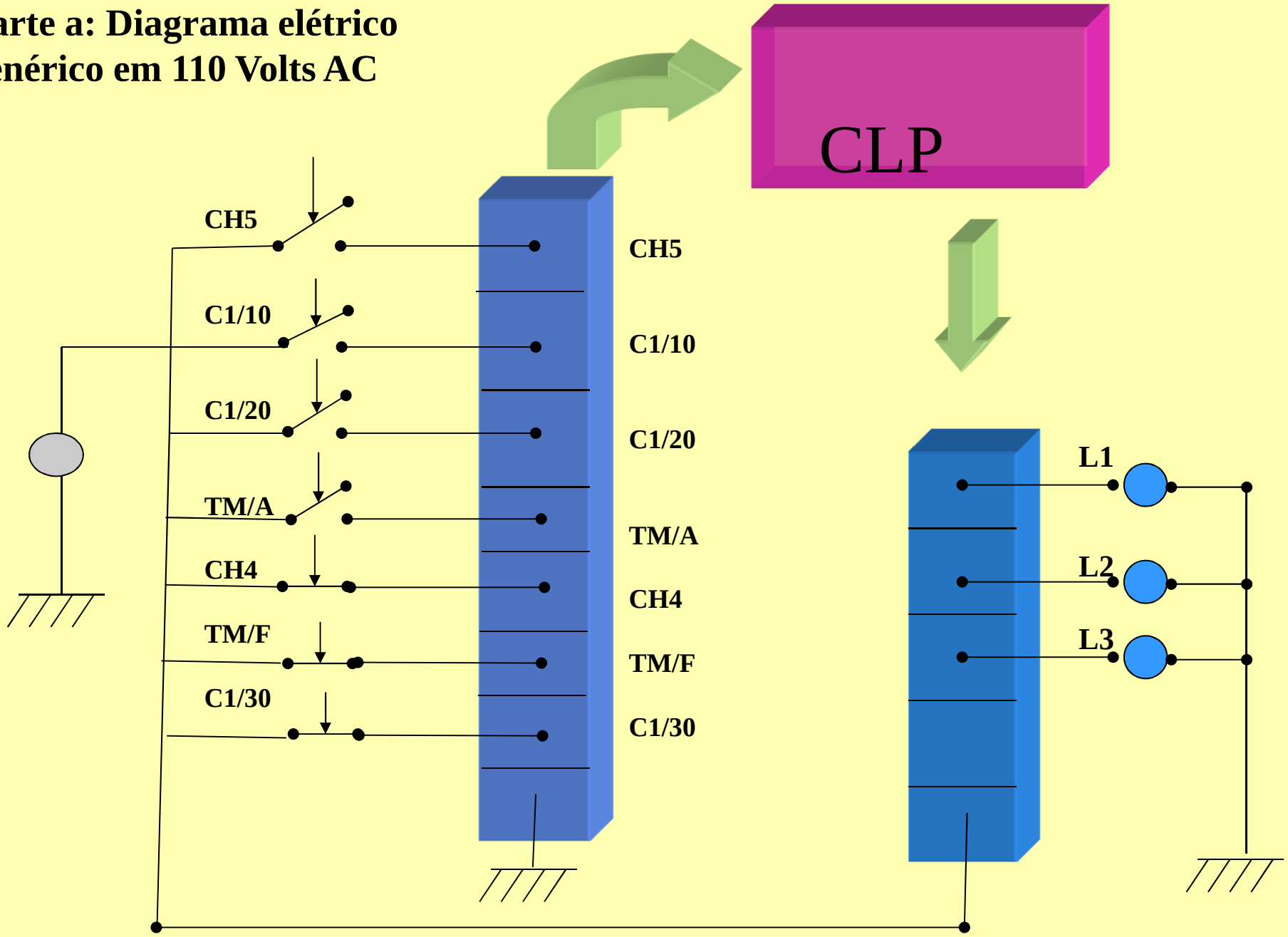
EX:1 Suponha o seguinte circuito elétrico de comando de um contator C1 onde CH4 é uma botoeira NF, CH5 é uma botoeira NA, TM é o relé térmico com dois contatos NA e NF e L1, L2 e L3 são sinalizadores luminosos.

a - Apresentar o circuito genérico de ligação elétrica de um CLP e os dispositivos descritos.

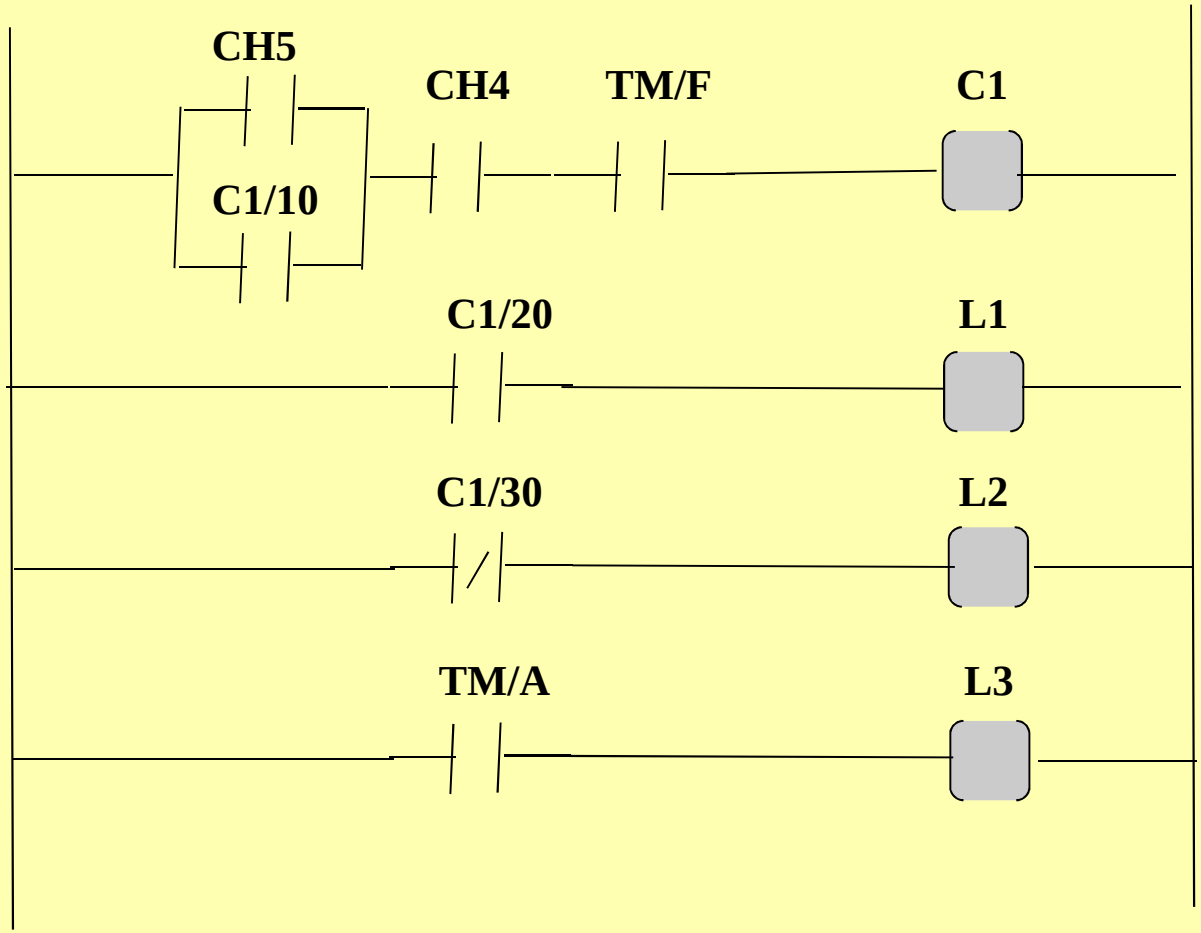
b - Apresentar um ladder genérico para comandar o contator C1 conforme o circuito elétrico o faz.



Parte a: Diagrama elétrico genérico em 110 Volts AC

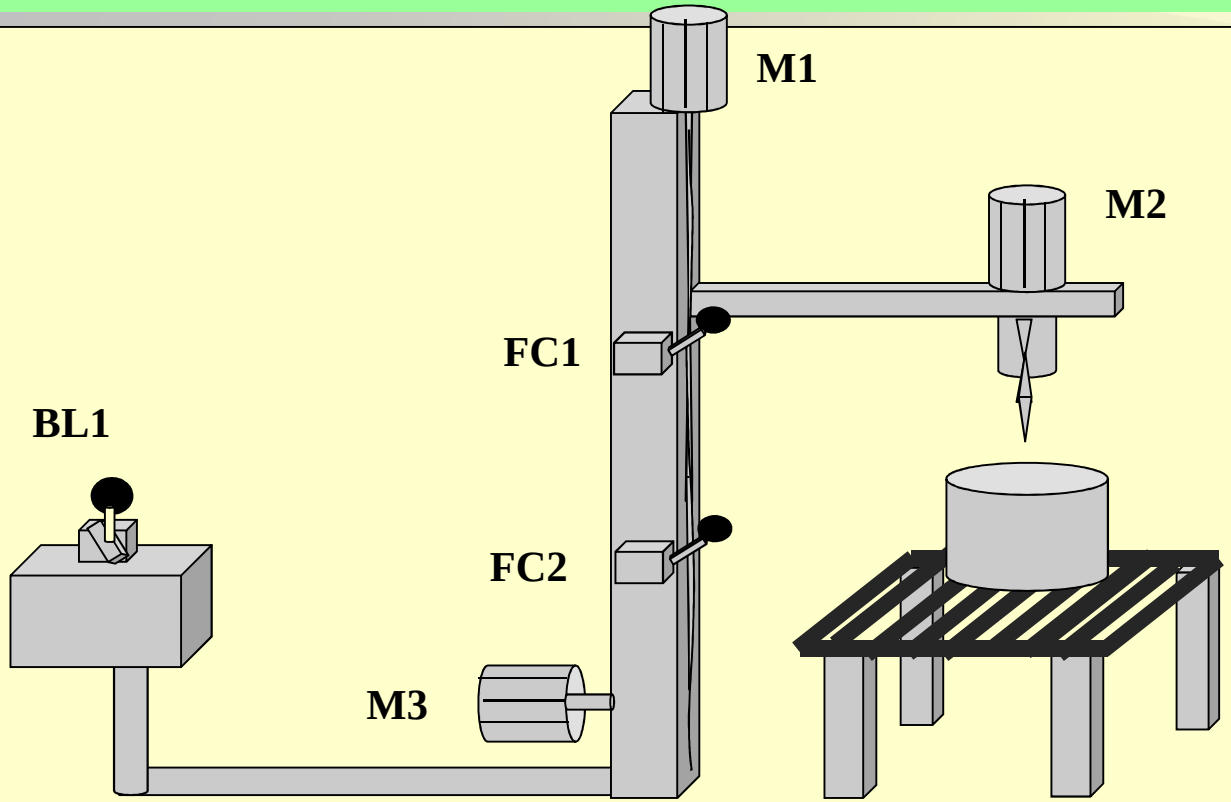


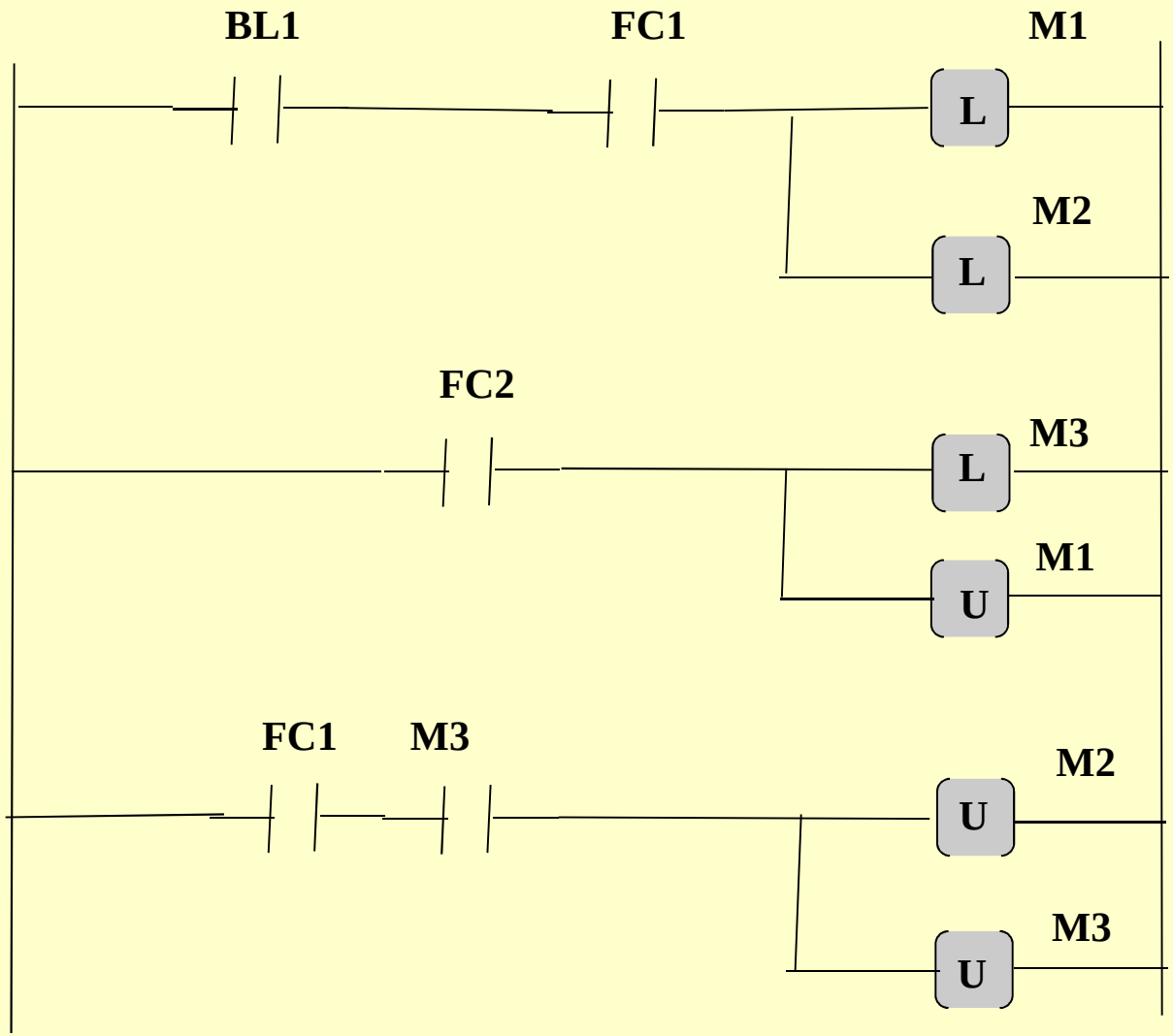
Parte b: Diagrama ladder genérico.



EX:2 Suponha uma furadeira com e os seguintes dispositivos:
uma botoeira NA (BL1) e dois sensores de fim de curso (FC1) e (FC2) do tipo NA. Existem três motores: M1 (descida), M2 (furadeira) e M3 (subida).
Fazer um diagrama ladder genérico para comandar a furadeira coforme o algoritmo seguinte: **(fonte: RA Treinamento Brasil 2013)**

- **Quanto FC1 está acionado e um pulso é dado no botão BL1, o motor M1 é ligado juntamente com o motor M2.**
- **Quando FC2 for acionado (estando M1 ligado e também o M2) deve-se desligar M1, manter M2 ligado e ligar o M3 (motor de subida).**
- **Quando FC1 for acionado (estando M2 ligado e também o M3) deve-se desligar M2 e M3.**



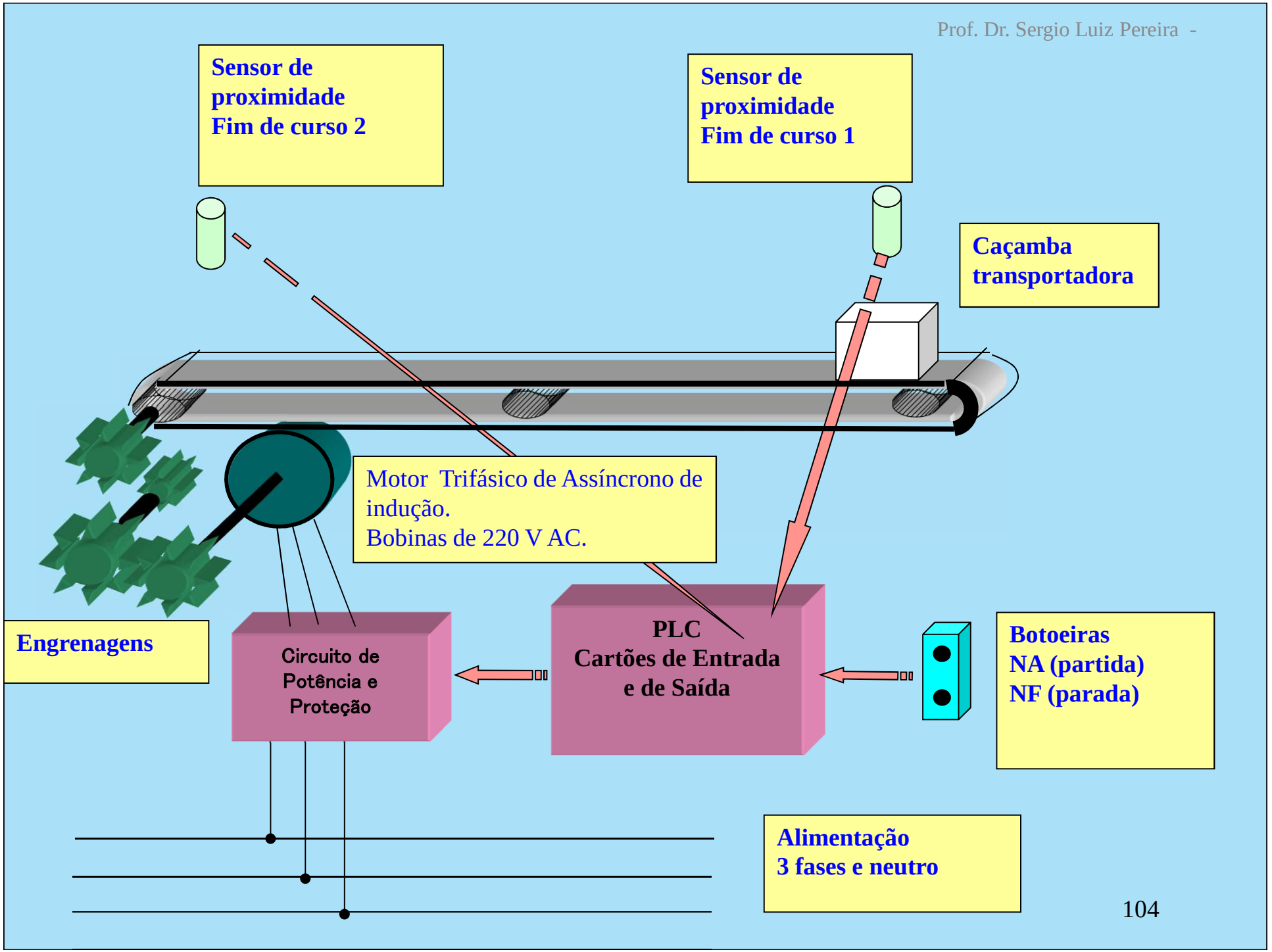

BL1
FC1
M1
L
M2
L
FC2
M3
L
M1
U
FC1
M3
M2
U
M3
U

Exercício 3: Automação Do Acionamento de Uma Esteira Transportadora. (Projeto do Circuito Elétrico e do Diagrama Ladder)

Deseja-se Automatizar o Acionamento de uma Determinada Esteira Transportadora.

A macro arquitetura de hardware é apresentada na figura seguinte.

Elaborar o Circuito Elétrico e o Diagrama Ladder para efetuar o algoritmo do processo descrito a seguir.


Sensor de proximidade Fim de curso 2
Sensor de proximidade Fim de curso 1
Caçamba transportadora
Motor Trifásico de Assíncrono de indução.
Bobinas de 220 V AC.
Engrenagens
Circuito de Potência e Proteção
PLC
Cartões de Entrada e de Saída
Botoeiras
NA (partida)
NF (parada)
Alimentação
3 fases e neutro

DESCRITIVO ALGORITMO DO PROCESSO DE TRANSPORTE

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

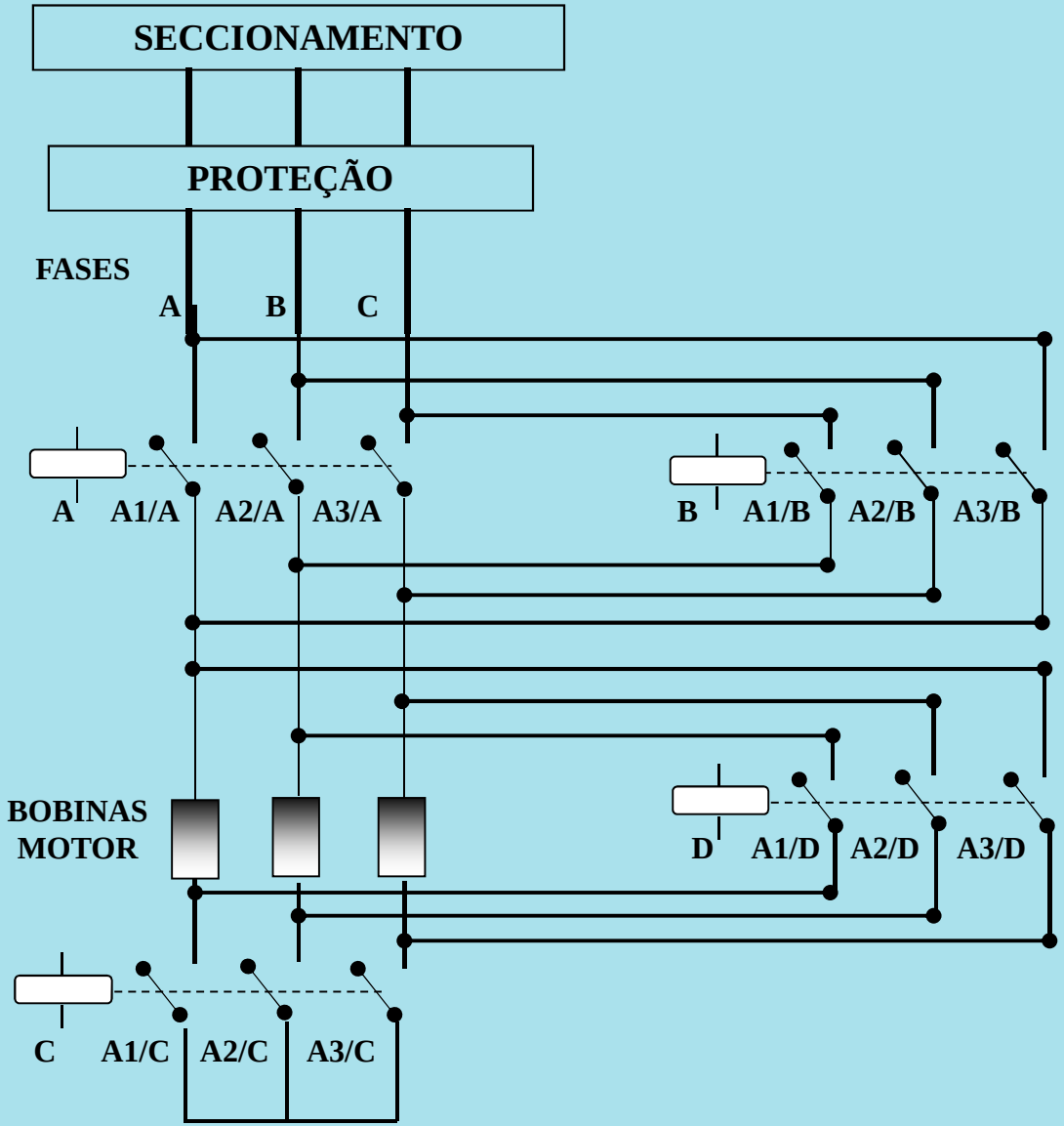
2) O motor deve ser acionado (ou seja, deve ser partido) com as bobinas ligadas em estrela e após cinco segundos a ligação das bobinas do estator deve ser automaticamente comutada para a ligação triângulo.

3) 0 motor irá acionar a esteira até que a caçamba de transporte de material atinja a posição 2 (esta situação é detectada pela chave fim de curso 2). 0 motor deve ser desligado por 15 segundos (tempo necessário (de simulação) para a carga/descarga do material) e em seguida a sua rotação deve ser automaticamente revertida de modo que a caçamba retome à posição 1.

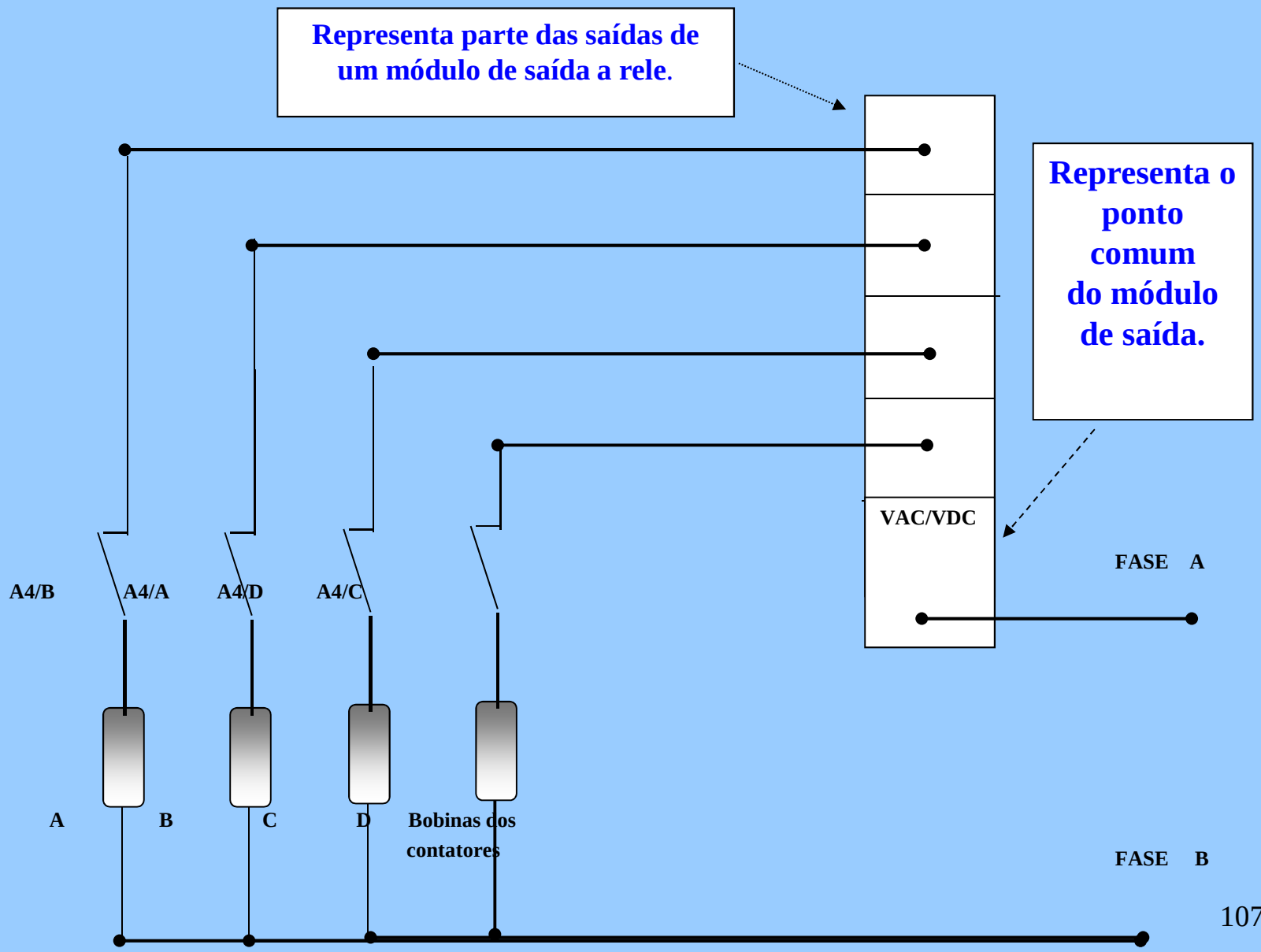
4) A botoeira de parada (do tipo normalmente fechada, "emergência") deve interromper o movimento da esteira em qualquer etapa do processo. Ao ser novamente acionada a botoeira de partida, o processo deve ser completado até o retorno da caçamba à posição 1.

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

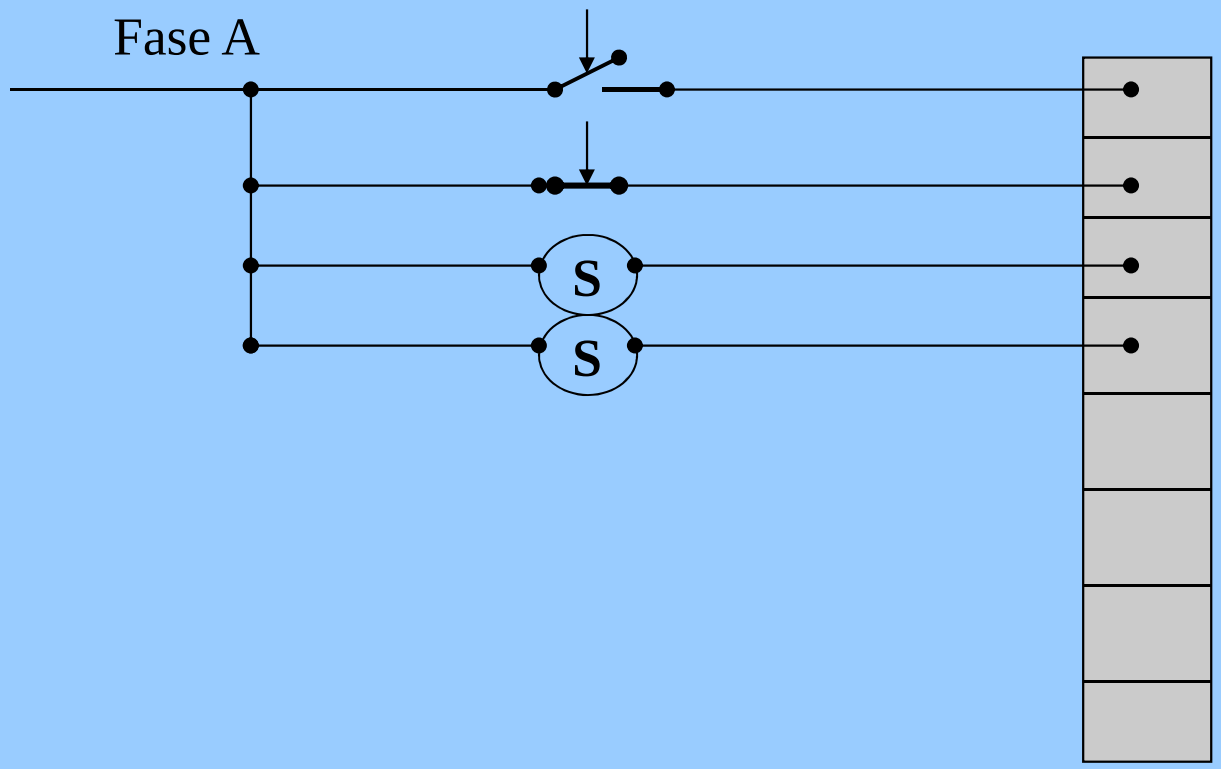
CIRCUITO DE POTÊNCIA

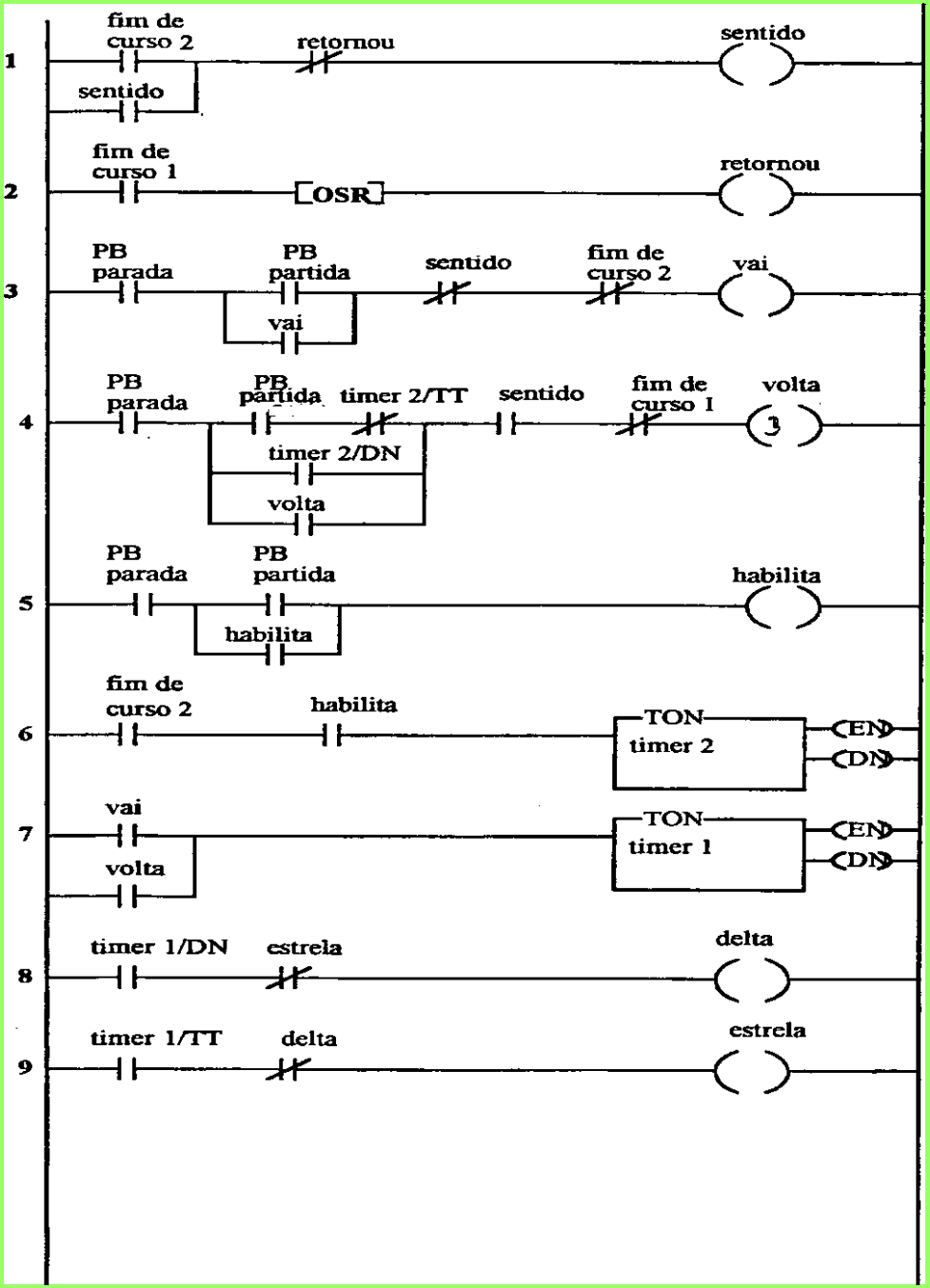


Ligação das bobinas com intertravamento de hardware por meio dos contatos auxiliares dos contatores



Ligação das botoeiras NA e NF e dos sensores de fim de curso





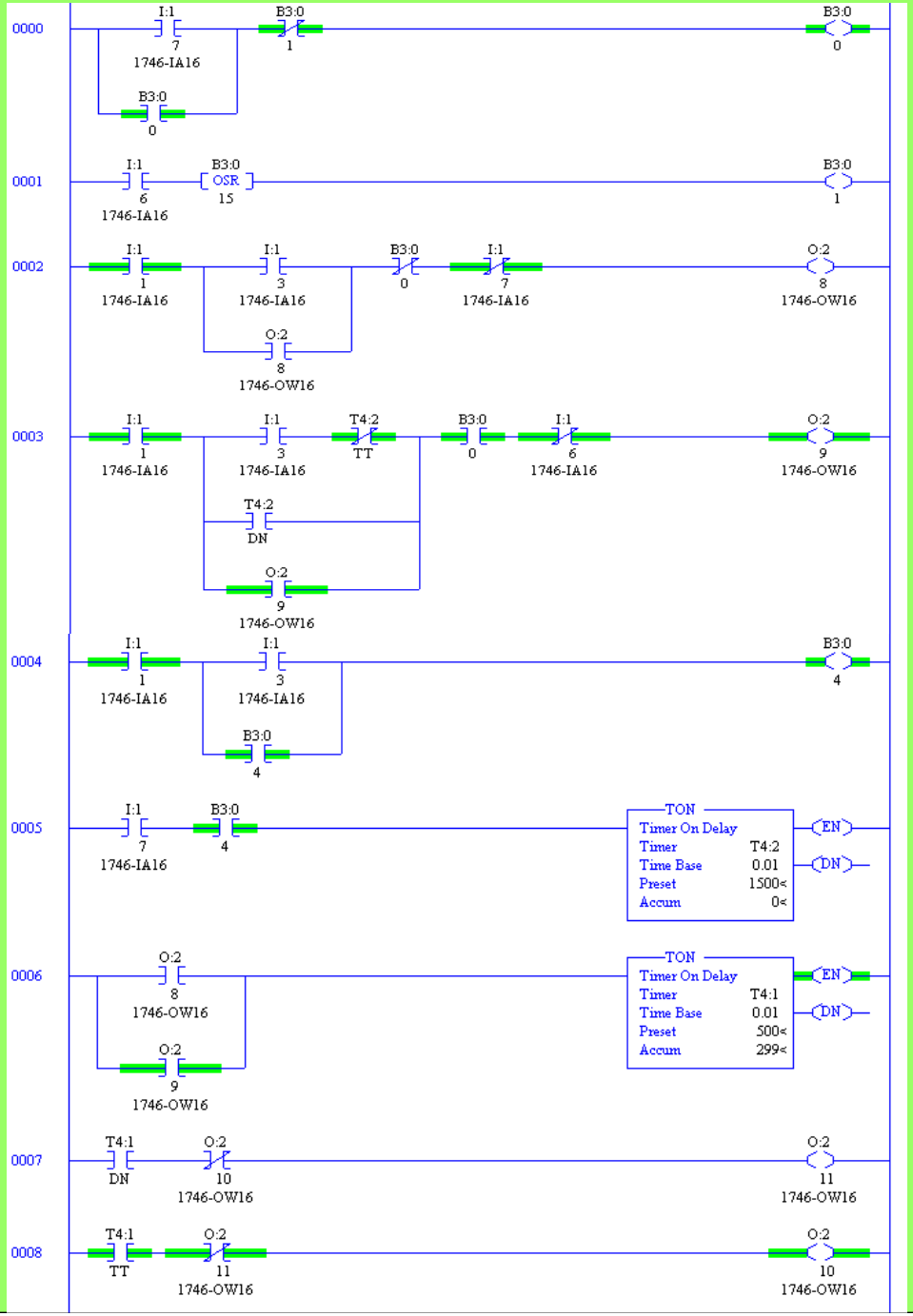
Moraes. C.; Senger, E; Pereira, S.L. Laboratório de Automação PEA - 1998

Exemplo prático de utilização das entradas com CLP SLC 500, Control_logix ou Micro_logix da Rockwell Automation.

Variável	saída	Endereço		Variável	Tipo	entrada	Endereço
Bobina A	**8**	**O:2/8**		**P.B. partida**	**NA**	**3**	**I:1/3**
Bobina B	**9**	**O:2/9**		**P.B. parada**	**NF**	**1**	**I:1/1**
Bobina C	**10**	**O:2/10**		**fim de curso 1**	**NA**	**6**	**I:1/6**
Bobina D	**11**	**O:2/11**		**fim de curso 2**	**NA**	**7**	**I:1/7**

Exemplo de Diagrama Ladder com CLP SLC 500 para implementar o algoritmo do processo.

(Pereira, S. L; Andrade, A. A. CLP - **Controladores Lógicos Programáveis. Laboratório de Automação Departamento de Engenharia de Energia e Automação Elétricas –** PEA – EPUSP 2005)



Controlador de Automação Programável (CAP)

CAPs são controladores que apresentam a elevada confiabilidade de hardware dos CLPs e a elevada capacidade de processamento e quantidade de memória dos microcomputadores.

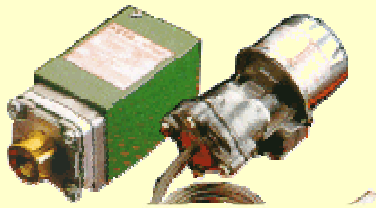
O objetivo principal no projeto de um CAP é que o mesmo possua as funções de um CLP e um SDCD (Sistema Discreto de Controle Distribuído com o preço e a simplicidade do CLP

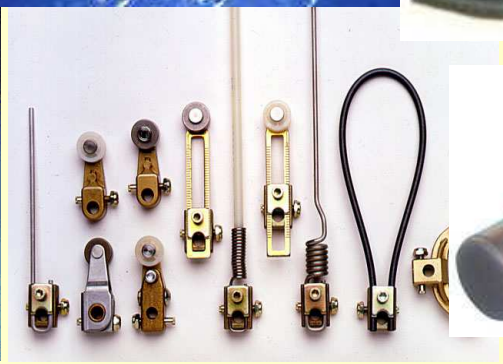
PAC (Programmable Automation Controller)

ou

CAP (Controlador de Automação Programável)

Controladores Programáveis





Sistemas Automatizados de Monit. e Controle

Equipamentos – Controladores - CLP

Os principais fabricantes e fornecedores de sistemas de automação industrial baseados em CLP's são:

- ***GE*** **– séries 90-70, 90-30 e VersaMax;**
- ***Rockwell Automation*** **– famílias PLC5, SLC-500 e MicroLogix;**
- ***Siemens*** **– famílias SIMATIC S7 e S5;**
- ***Schneider*** **– famílias Modicon M340, Premium, Quantum e TSX Micro.**

Os principais fabricantes e fornecedores de sistemas de automação industrial baseados em PAC's são:

GE **– séries RX7i e RX3i ;**

Rockwell Automation **– família ControlLogix ;**

Siemens **– família SIMATIC S7;**

Schneider **– família Modicon M340;**

ABB **– sistema Compact Products 800.**

Os principais fabricantes e fornecedores de sistemas de automação industrial baseados em SDCD's são:

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

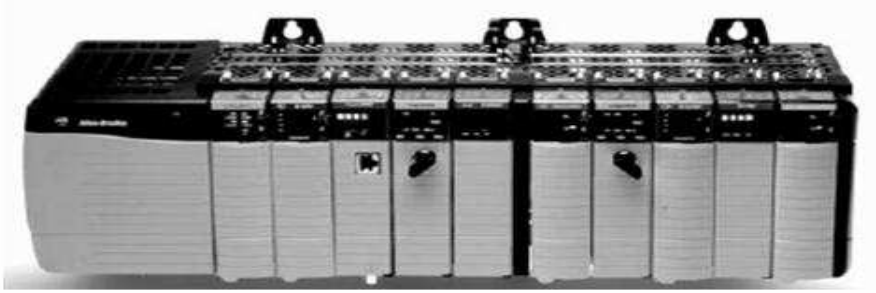
b - Emerson – sistema DeltaV;

c - Honeywell – sistemas Experion e PlantScape;

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

e – Siemens

CAP ControlLogix



Memória de dados e lógica

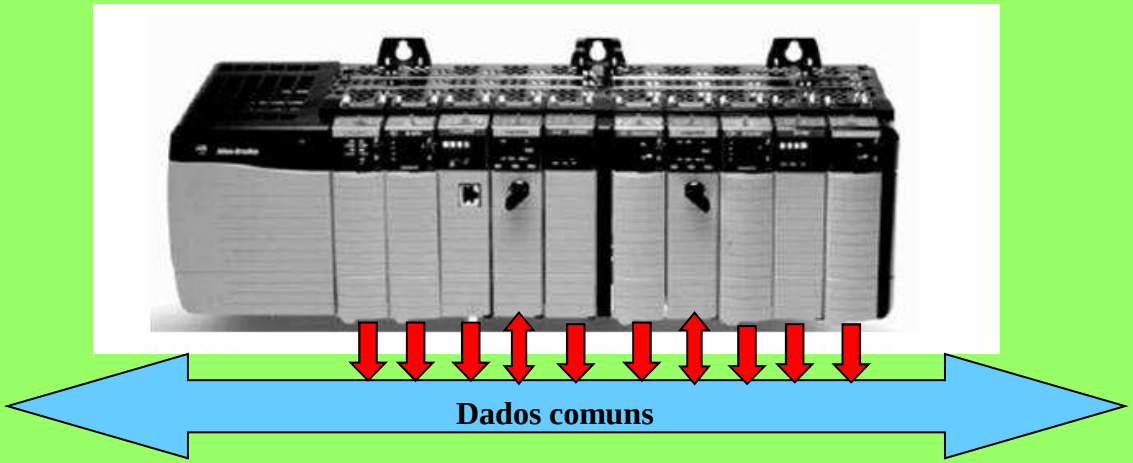
Código fonte do Programa
Código fonte do Programa
Código fonte do Programa

UCP Logix

Memória de Entradas/Saídas

Dado de entradas/Saídas
Tabela de E/S forçada
“Buffers” de mensagem
Tags Produtor/Consumidor

UCP Backplane

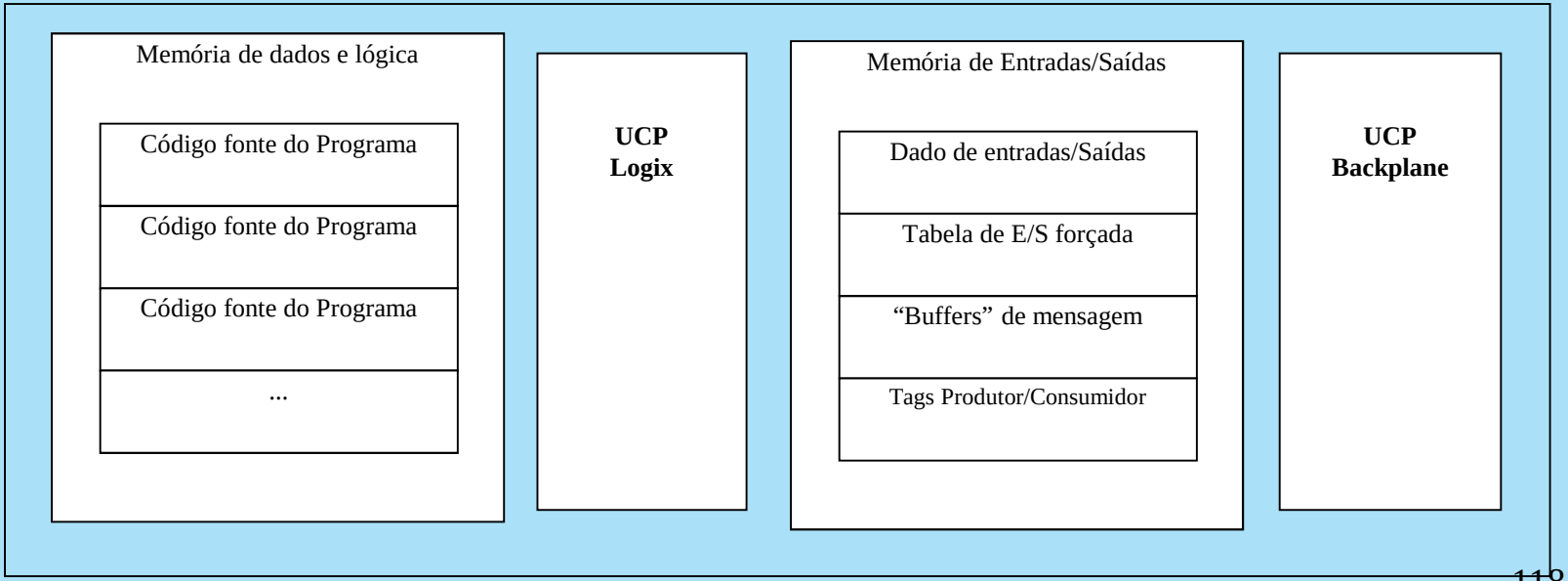


Na arquitetura do CLX existem pelo menos duas UCPs: a denominada UCP Logix e a UCP de "backplane". *Backplane* é a placa onde tanto os CAPs, CLPs e cartões de Entrada/Saída e de comunicação são conectados. para o controlador mostrado, a "backplane" é a placa traseira do rack.

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

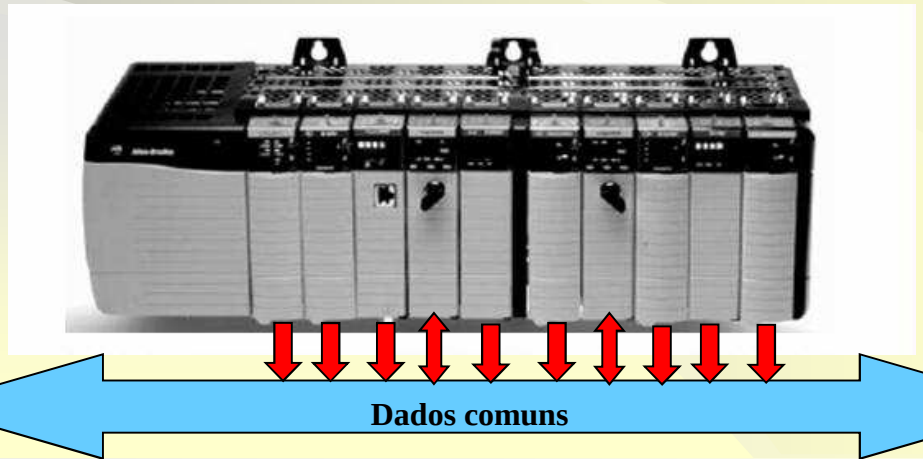
A UCP de "backplane" se comunica com os cartões de entrada e saída, e opera de forma independente da UCP do Logix.

Esta arquitetura permite uma maior versatilidade de operação, uma vez que as duas UCPs operam de forma assíncrona e independente.

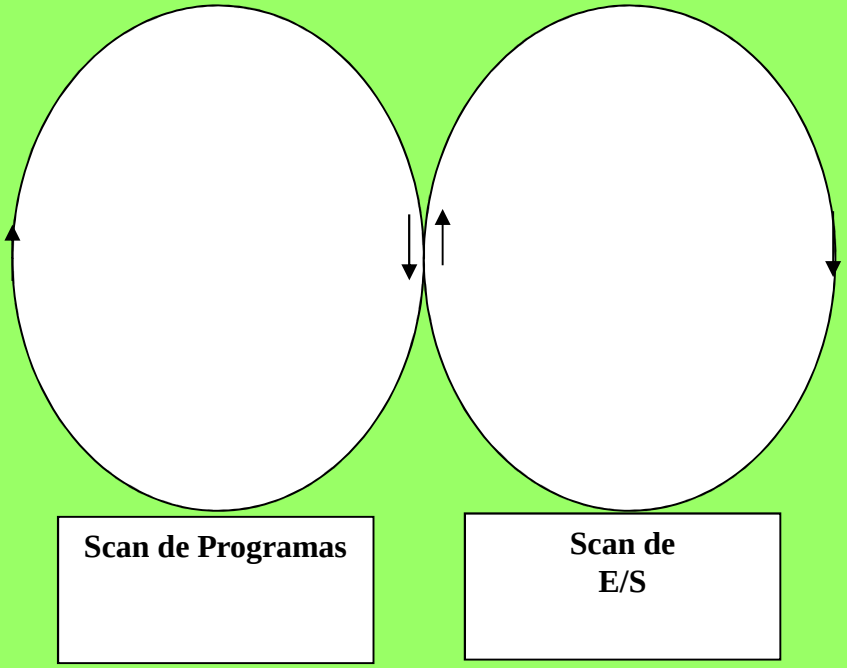


A placa de fundo chassi "backplane" opera baseada na rede ControlNet, por onde trafegam dados (pelo barramento de dados) e também existe o barramento de alimentação do rack, cartões e CPU. A rede trabalha no modelo Produtor/Consumidor. ("Alguém gera informação para quem desejar empregá-la")

Isto permite que a CPU fique em qualquer lugar do rack e que também haja mais do que uma CPU no mesmo rack.



ControlLogix existem dois tipos de Scan assíncronos: O Scan de programa (de cada controlador) e o Scan de entradas e saídas.



Obs. Para os módulos de saída, independente do RPI, há sempre uma atualização das saídas ao final do Scan cada programa. Isto somente é válido para cartões locados no mesmo chassis da UCP.

Ilustrativo dos dois tipos de Scan na arquitetura ControlLogix!

CONEXÕES: (Até 250 no V.6 e 500 no V.7)

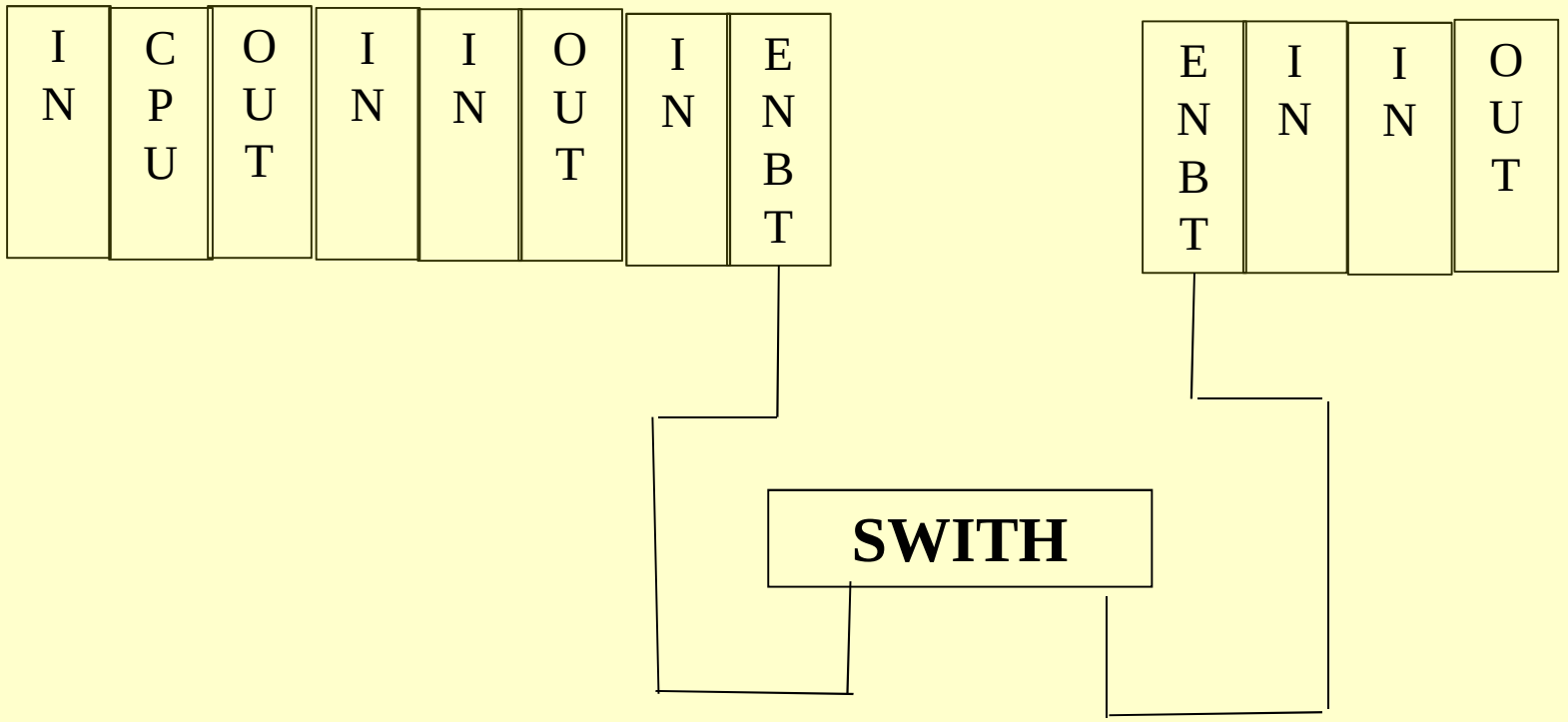
Conexões: É um enlace entre dois dispositivos. Estes dispositivos podem ser controladores, módulos de comunicação, módulos de E/S, variáveis produzidas e consumidas ou mensagens. O ContrlLogix suporta até 250 conexões:

Conexão Direta: É quando cada cartão consome uma conexão.

Conexão Rack Otimizado: è quando cada rack consome apenas uma conexão.

Conexão Mista: (direta mais rack otimizado): É quando existem módulos analógicos no rack remoto, porque cada cartão analógico obrigatoriamente consome uma conexão.

CONEXÕES: (Até 250 no V.6 e 500 no V.7)



- **Dos controladores para os módulos E/S local como remoto.**
- **Do s controladores para os módulos de comunicação local ou remoto.**
- **Entre variáveis produzidas e consumidas.**
- **Através de mensagens.**

Atualização das Entradas e Saídas:	**A troca de dados entre dispositivos de E/S e o Controlador obedece ao modelo produtor/consumidor. Desta forma a varredura de entrada e atualização das saídas não está necessariamente atrelada ao Scan.**
RPI – Request Packer Interval:	**Especifica a taxa na qual os dados são produzidos por um cartão de entrada ou saída. Este valor está compreendido entre 0,2 a 750 mili segundos.**
COS – Change of Sate Somente para módulos digitais:	**Um módulo/cartão de entrada produzirá informação somente quando houver uma transição de On para OFF ou OFF para ON, detectada pelo circuito de entrada.**

RTS – Real Time Sample Somente para módulos analógicos de entrada:	**É o tempo gasto para executar as seguintes ações: ler todos os canais de entrada, atualizar status e zerar o contador do RTS.**
Endereçamento:	**No ControlLogix não existe uma tabela de E/S pré-definida. Esta tabela é formada conforme os módulos são configurados.** **Endereçamento local: é quando o módulo está No mesmo rack que a UCP do ControlLogix.** **Endereçamento Remoto: é quando o módulo não está no mesmo rack que a UCP do ControLogix.**

Compartilhamento de E/S: Os cartões de E/S do ControlLogix podem ser compartilhados de três formas:

Multicast: Mais de uma UCP pode ser proprietária de um único cartão: Este compartilhamento funciona apenas para cartões de entrada e que esteja configurados da mesma forma em todas as UCPs.

Owner: Quando somente uma única UCP pode ser proprietária de um cartão. Os cartões de saída só podem ser configurados em uma UCP como proprietário, pois não é possível que duas UCPs escrevam no mesmo cartão de saída.

Listen Only: Quando uma UCP somente pode ter os dados de um cartão, não podendo escrever nem ser configurado mesmo. Um cartão de saída deve ser configurado em uma UCP como owner, mas nas outras o cartão deve ser configurado como listen Only.

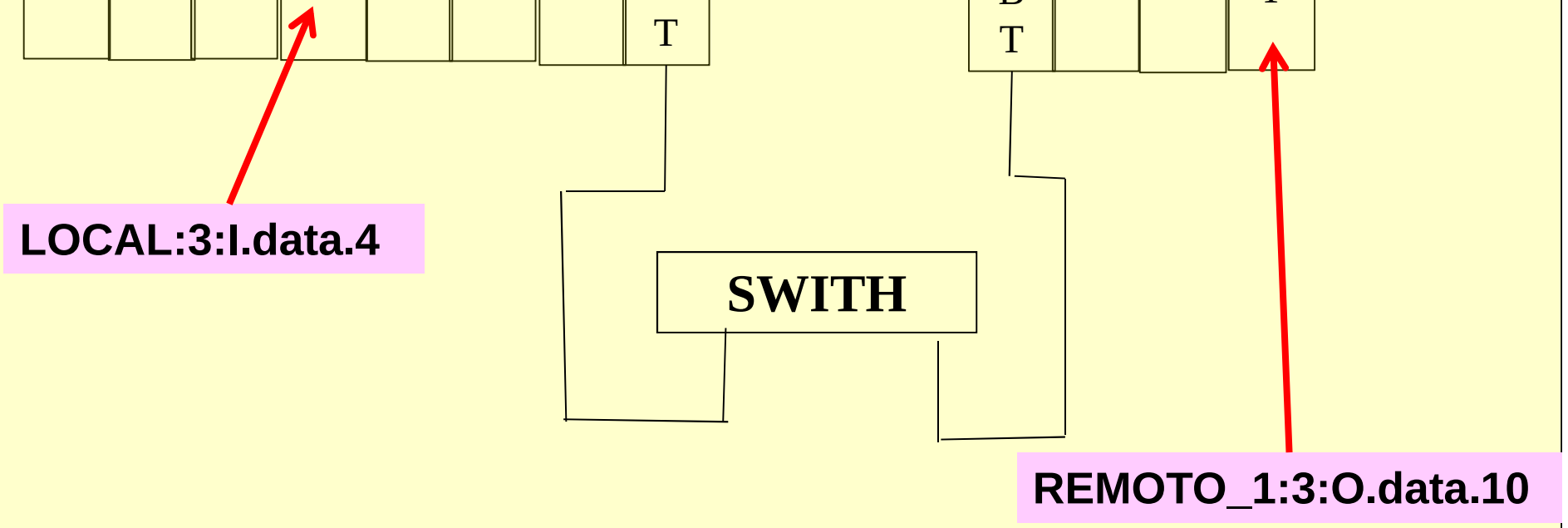
Endereçamento: (O cartão de comunicação é quem determina o nome do chassi remoto)

Local

00	01	02	03	04	05	06	07
IN	CPU	OUT	IN	IN	OUT	IN	ENBT

Remoto

00	01	02	03
ENBT	IN	IN	OUT



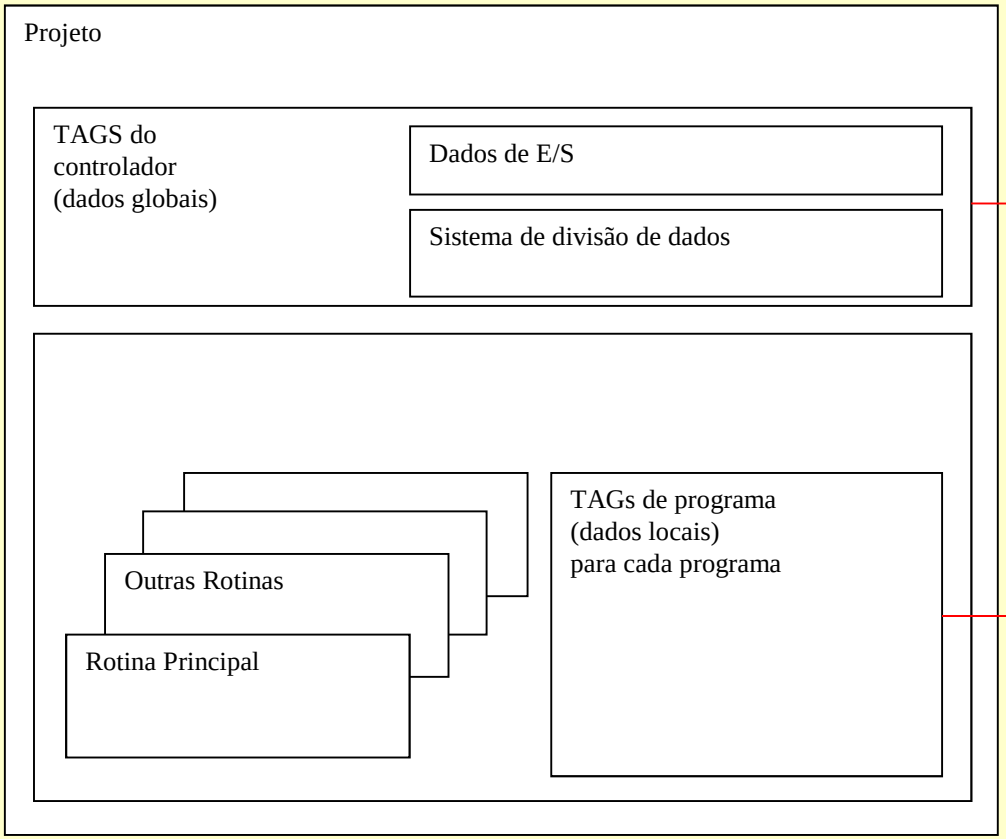
ESTRUTURA:

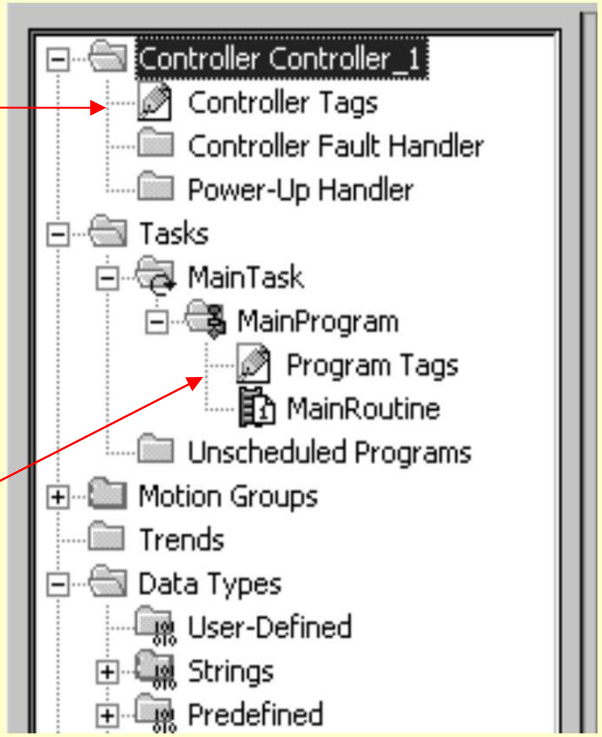
LOCALIZAÇÃO:Slot:Tipo.Membro.Submembro.Bit

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

UM PROJETO :

É a aplicação completa. É o arquivo que armazena: a lógica, configurações, dados e a documentação para o controlador.





Projeto	É a aplicação completa. É o arquivo que armazena: a lógica, configurações, dados e a documentação para o controlador.
Tarefas (Tasks) 1 contínua e até 31 Periódicas o de Eventos	Uma tarefa (task) é o mecanismo de escala de execução de um programa. Uma aplicação pode ser dividida em muitas (tasks). As (tasks) possibilitam que haja uma escala de operação e também de prioridade das tarefas a serem executadas pelo controlador. Existem três tipos de tasks: a – Tarefas (Tasks) contínuas: Estas (tasks) são executadas continuamente a menos que uma (tasks) periódica ou baseada em evento seja acionada. b – Tarefas (Tasks) periódicas: São (tasks) que são executadas em intervalos de tempo definidos. A taxa de uma (task) periódica pode ser de 0,1ms à 2.000 segundos. (prioridade de 1 á 15) c – Tarefas (Tasks) baseadas em eventos: São executadas apenas quando um evento especificado ocorre.
Programa	Um programa pode ser definido como um conjunto relacionado de rotinas e tags. Um programa possui uma ou mais rotinas ou sub–rotinas.
Rotinas	É um conjunto de instruções lógicas escrito em uma das linguagens de programação.

IMPORTANTE:

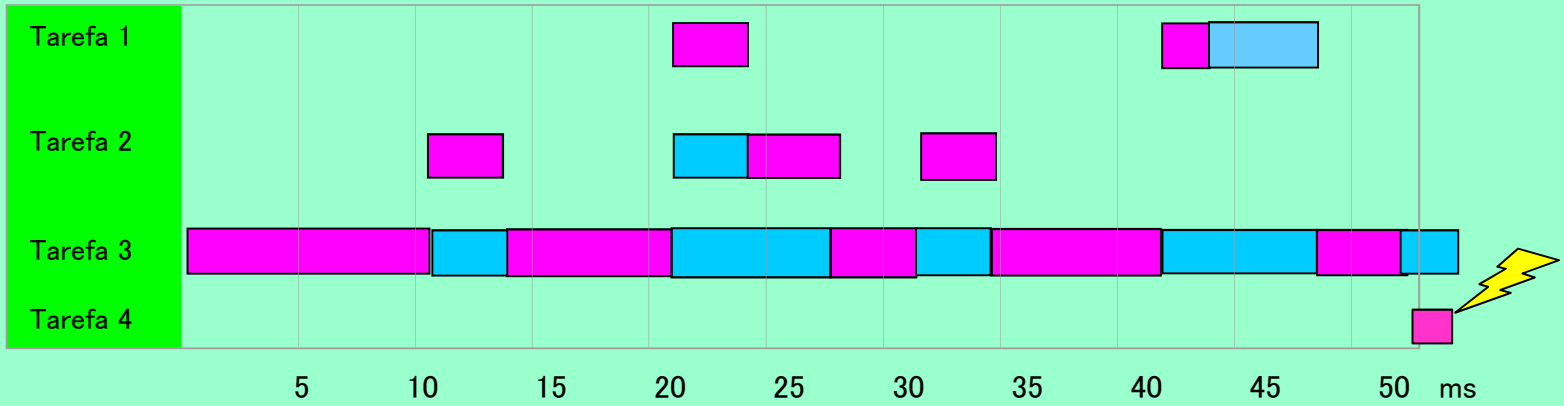
a – Tarefas (Tasks) contínuas: Estas (tasks) são executadas continuamente a menos que uma (tasks) periódica ou baseada em evento seja acionada.

b – Tarefas (Tasks) periódicas: São (tasks) que são executadas em intervalos de tempo definidos. A taxa de uma (task) periódica pode ser de 0,1ms à 2.000 segundos. (prioridade de 1 á 15)

c – Tarefas (Tasks) baseadas em eventos: São executadas apenas quando um evento especificado ocorre.

Exemplo 1 de execução de tarefas em função do tipo e da prioridade.

Tarefa	Tipo de Tarefa	Nível de prioridade	Tempo de Execução
1	Periódica de 10 ms	5	2 ms
2	Periódica de 20 ms	10	4 ms
3	Contínua	nenhum	24 ms



Legenda

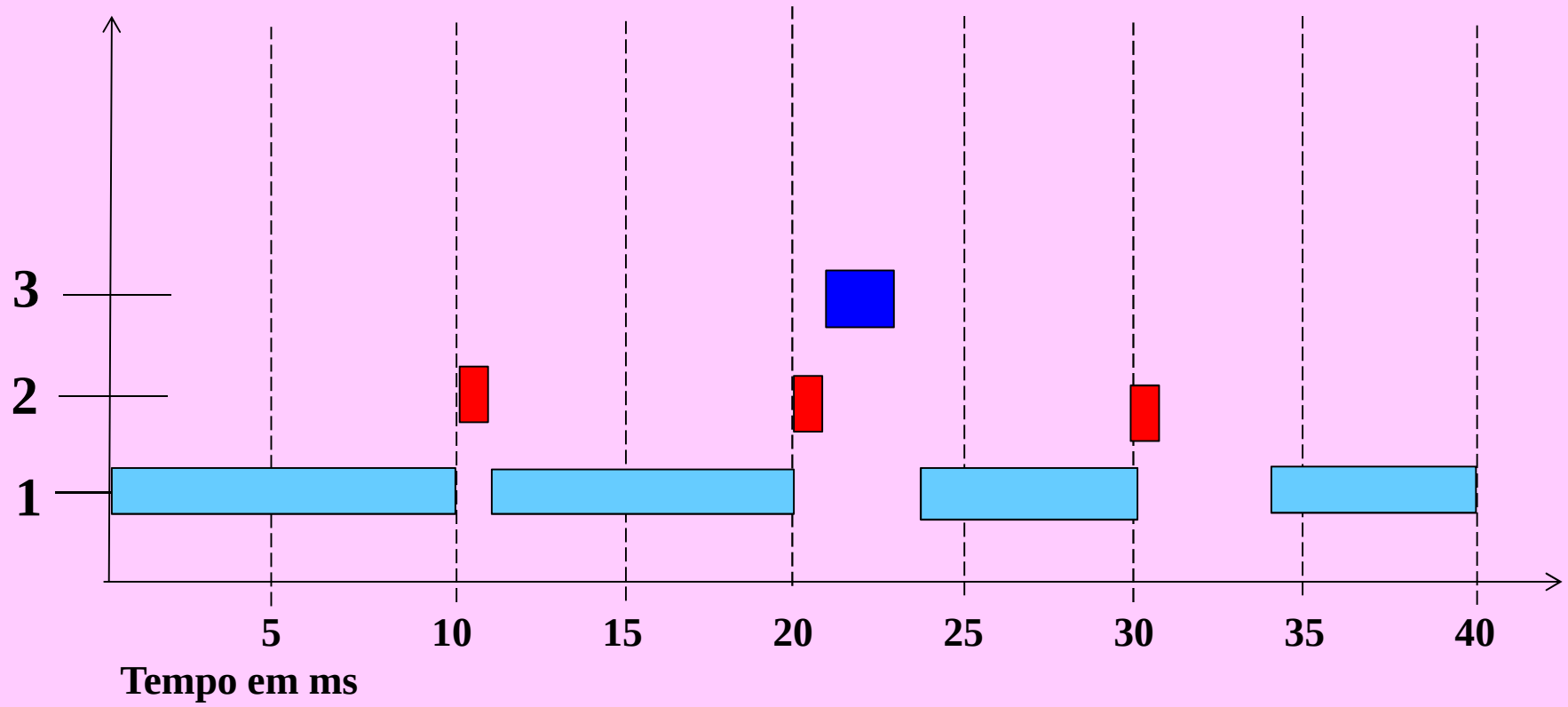
Tarefa é executada

Tarefa é interrompida

Ocorrência de um evento para tarefa 4

Exemplo 2:

Tarefas	Tipo	Tempo de execução	Prioridade
1	Contínua	24 ms	
2	Periódica	10 ms	5
3	Periódica	20 ms	10



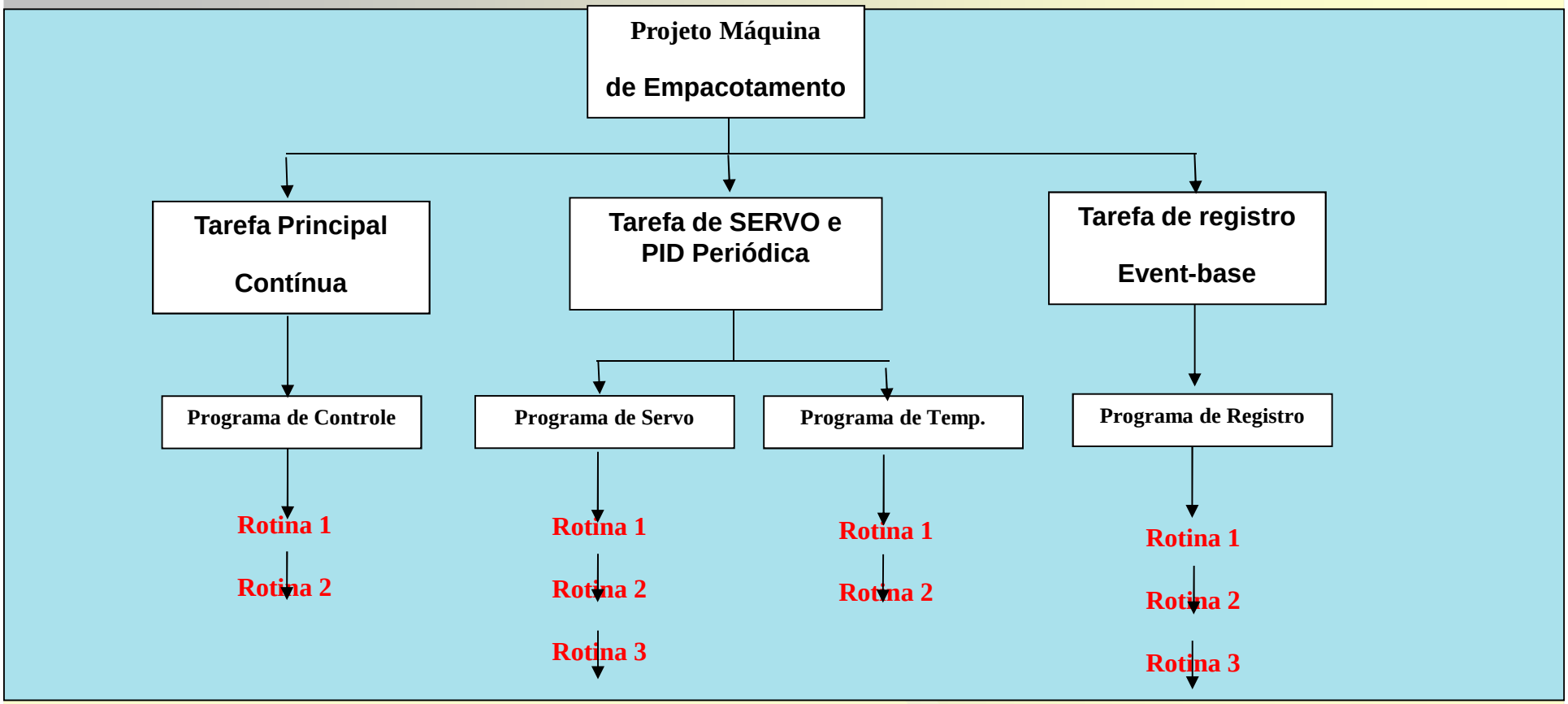
Deve-se observar que:

a – Todas as tarefas periódicas interrompem a tarefa contínua.

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

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

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



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

Endereçamento no Controlador de Automação Programável (CAP) ControlLogix (CLX) (TAG endereçamento): TAGs, Alias, Array e ADD On Instruction.

No CLX o endereçamento das entradas, saídas, bits, variáveis internas, temporizadores, contadores são efetuados por meio de TAGs. O processador emprega o nome da TAG para acessar o endereço do dado.

O nome da TAG identifica o dado e isto permite que a documentação do programa seja efetuada de forma clara na representação da aplicação.

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

Observe-se que a memória mínima empregada por uma TAG são 4 bytes ou 32 bits para o dado em si e mais 40 bytes para o nome da própria TAG. Em um projeto com CLX existem quatro tipos de TAGs: Base, Alias, Produced e Consumed. A tabela 4 ilustra os tipos de TAGs.

Tabela - Tipos de TAGs no CLX.

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

Uma Base TAG armazena qualquer tipo de dado para ser empregado na lógica do projeto, tais como: bit, inteiro, etc. A tabela 5 ilustra os tipos de Base TAGs e os respectivos empregos para cada tipo.

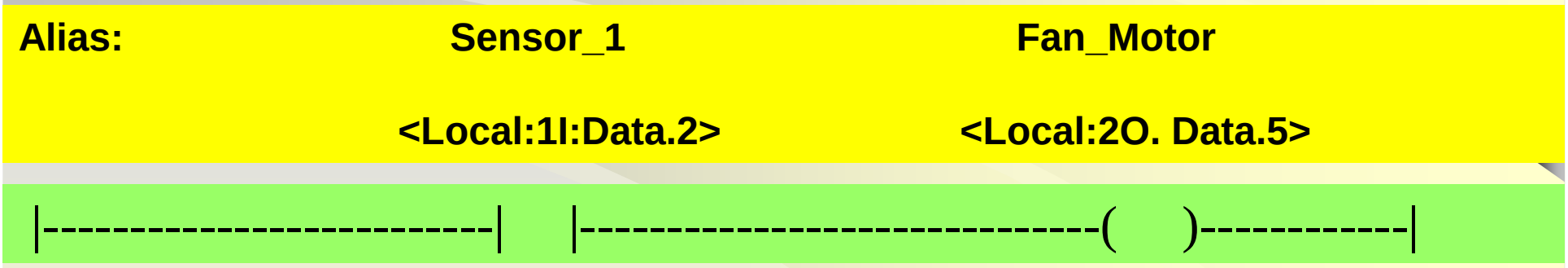
Tabela - Tipos de Base TAGs e os respectivos empregos para cada tipo.

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

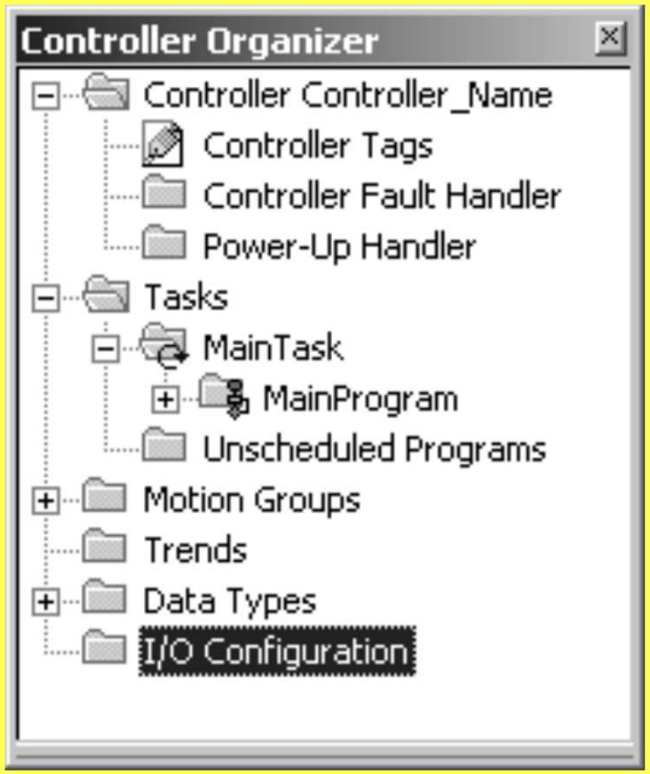
Tabela – Exemplo dos bits empregados no armazenamento dos valores para cada tipo de Base TAG.

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

Uma TAG tipo Alias (Apelido) é usada para criar um nome alternativo, ou seja um "apelido" para uma TAG já existente e já nomeada. Um Alias é um símbolo para um endereço específico de entrada ou saída. Esse nome pode representar uma entrada ou saída do mundo real, ou seja, a entrada ou saída de um dispositivo de campo. Um Alias também pode ser interpretado como uma TAG indexada nela mesma, ou seja, ele está "linkado" ou relacionado a TAG base. Assim, qualquer ação na TAG base também acontece com o Alias e vice e versa.



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



Location **Slot** **Type** **Member** **SubMember** **Bit**

... **Optional**

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

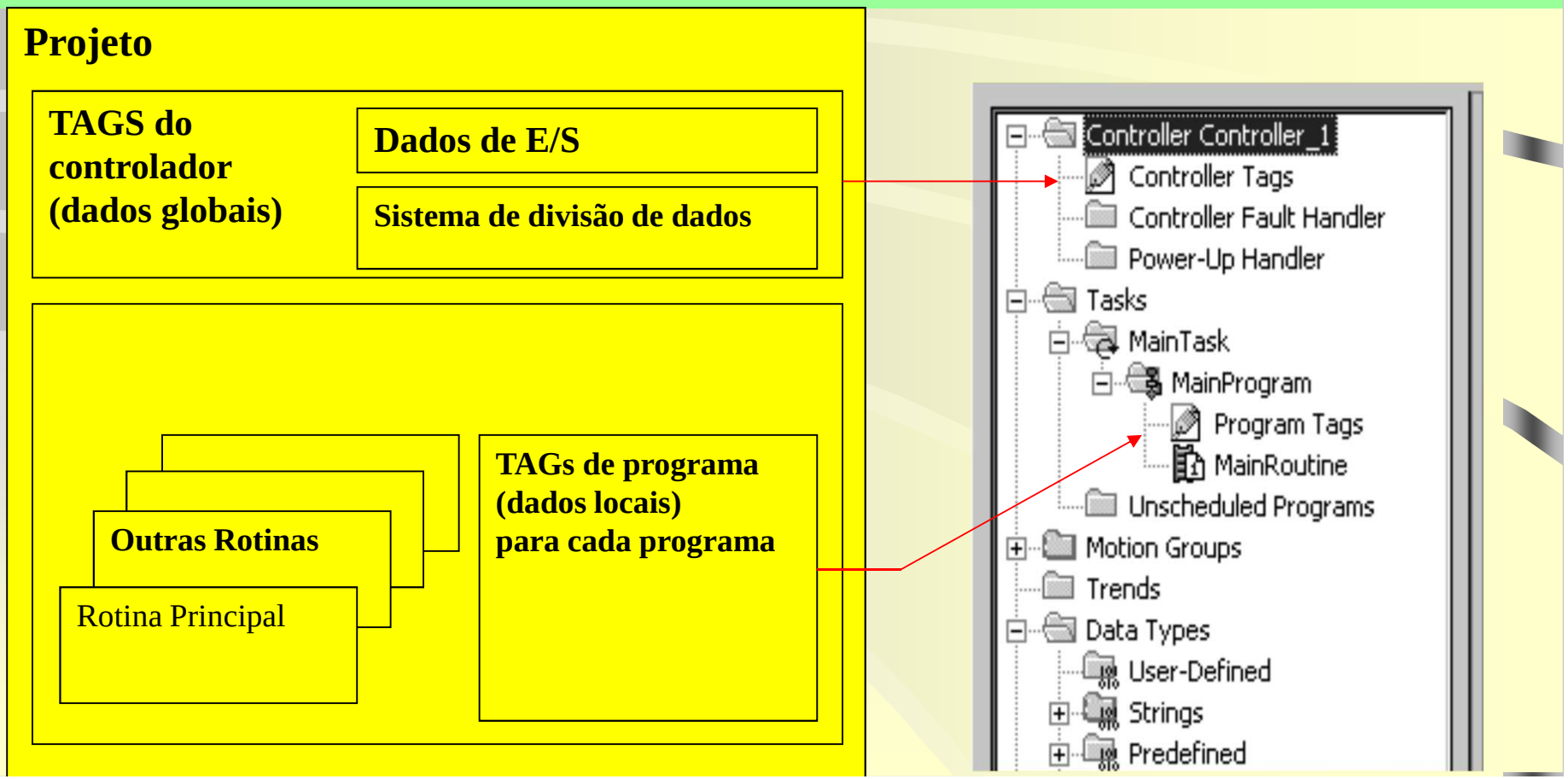
Where	Is
Location	Network location. LOCAL = same chassis or DIN rail as the controller ADAPTER_NAME = identifies remote comm. adapter or bridge module
Slot	Slot number of I/O module in its chassis or DIN rail
Type	Type of data: I = input O = output C = configuration S = status
Member	Specific data from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

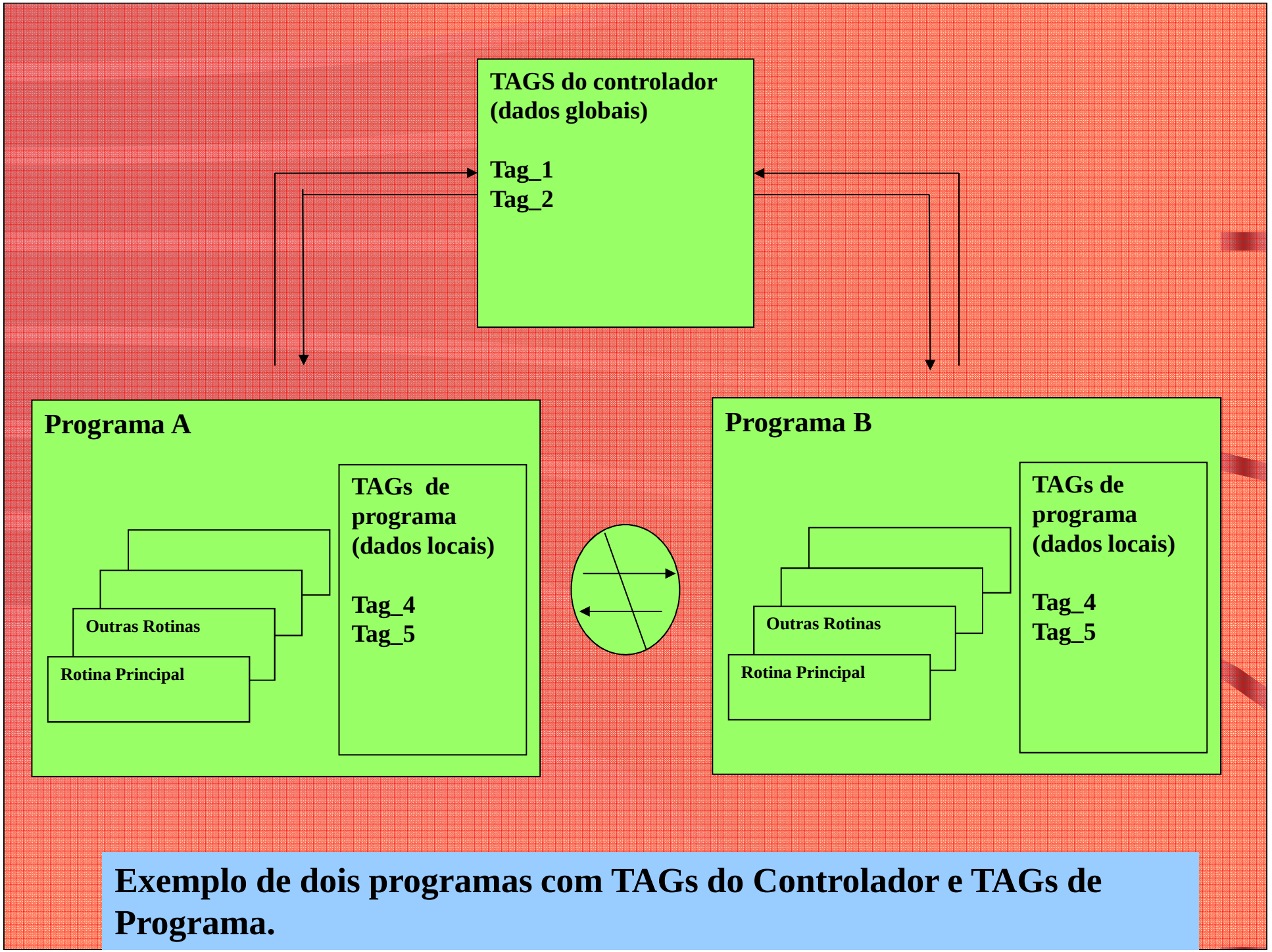
Um Array é um tipo de TAG que contém um bloco de muitos pedaços de dados. Um Array é similar a uma tabela de valores. Com um array de valores de dados, cada pedaço do dado individual é denominado elemento. Cada elemento de um array é do mesmo tipo de dado do restante dos elementos. Um array é uma matriz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

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

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

•TAG de Programa: as TAGs de escopo de programa "program scope TAGs" estão disponíveis apenas dentro dos programas em que foram criadas.





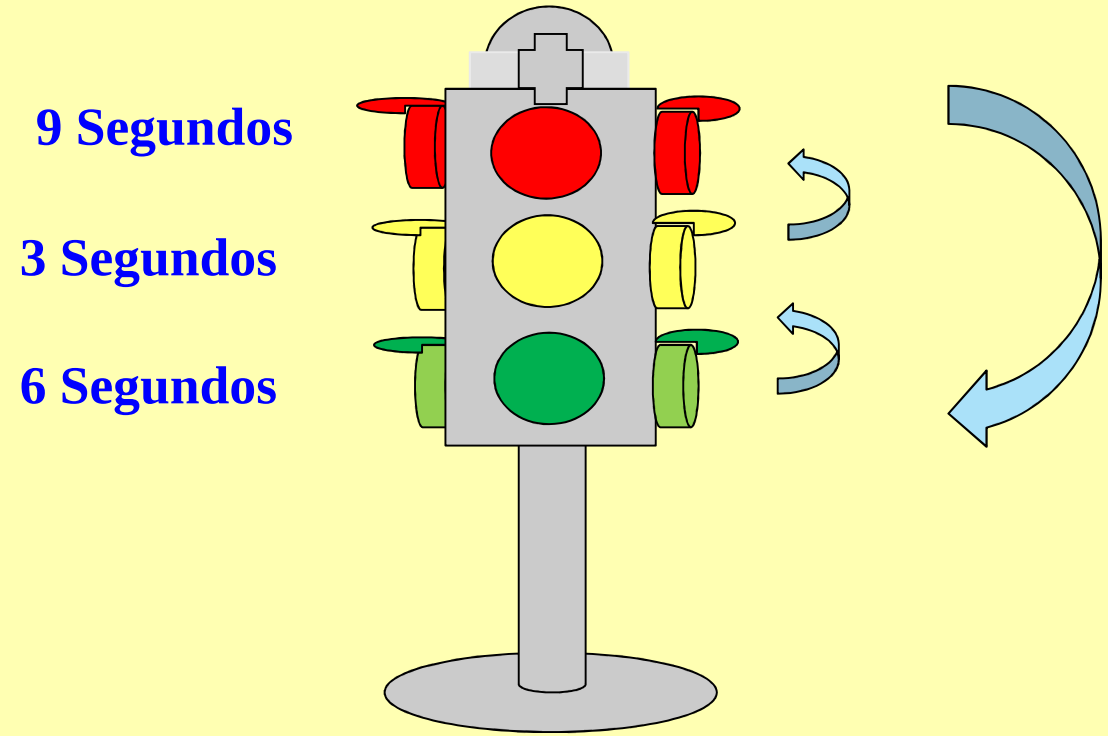
Exemplo de dois programas com TAGs do Controlador e TAGs de Programa.

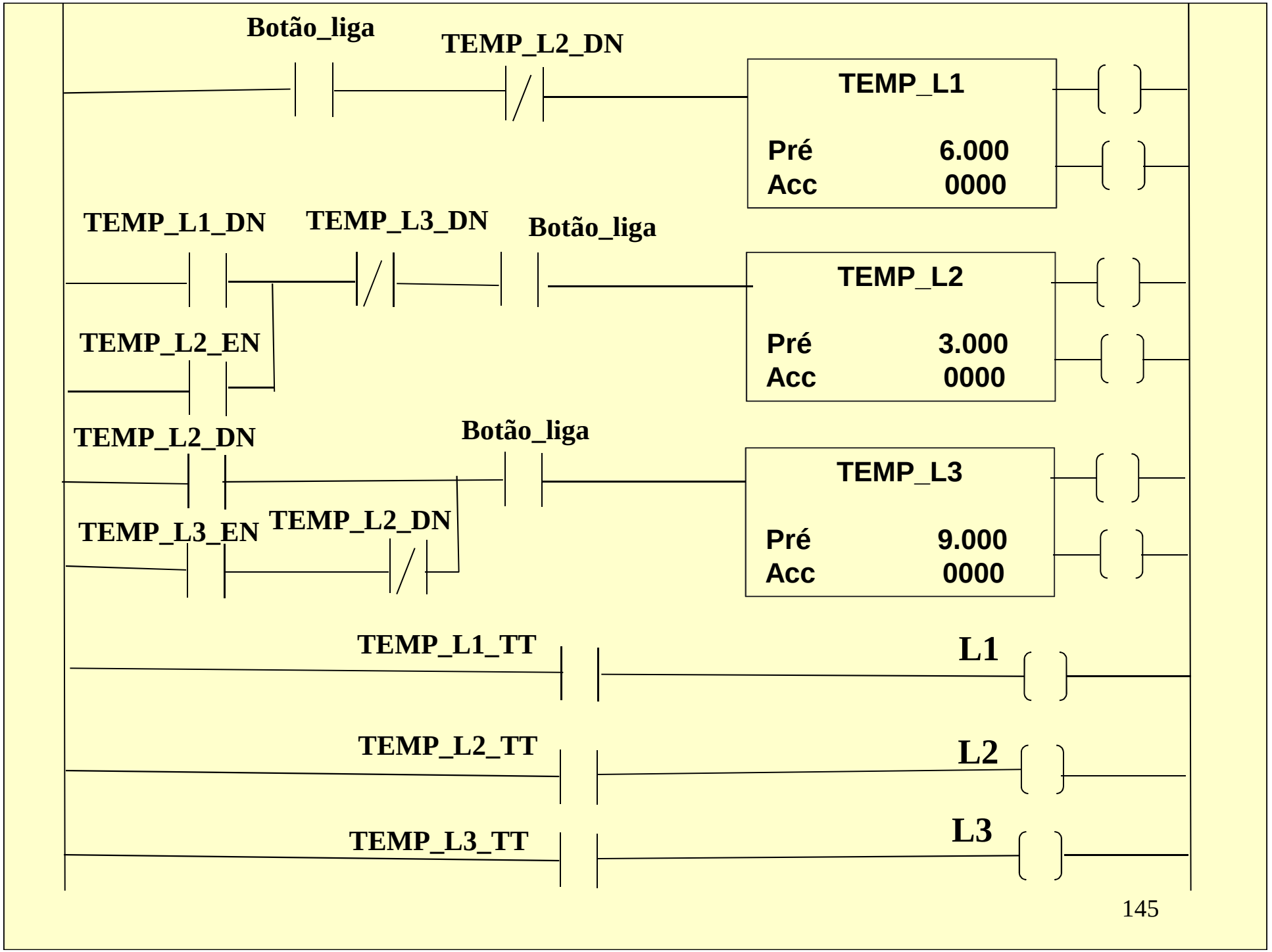
Exercício 4: Semáforo

1 - Criar uma rotina com o nome semáforo.

2 - Dentro do arquivo semáforo criar um programa de tal forma que ao acionar a chave retentiva do simulador , o semáforo energize suas lâmpadas na sequência indicada pelas setas e nos intervalos de tempo indicados na figura.

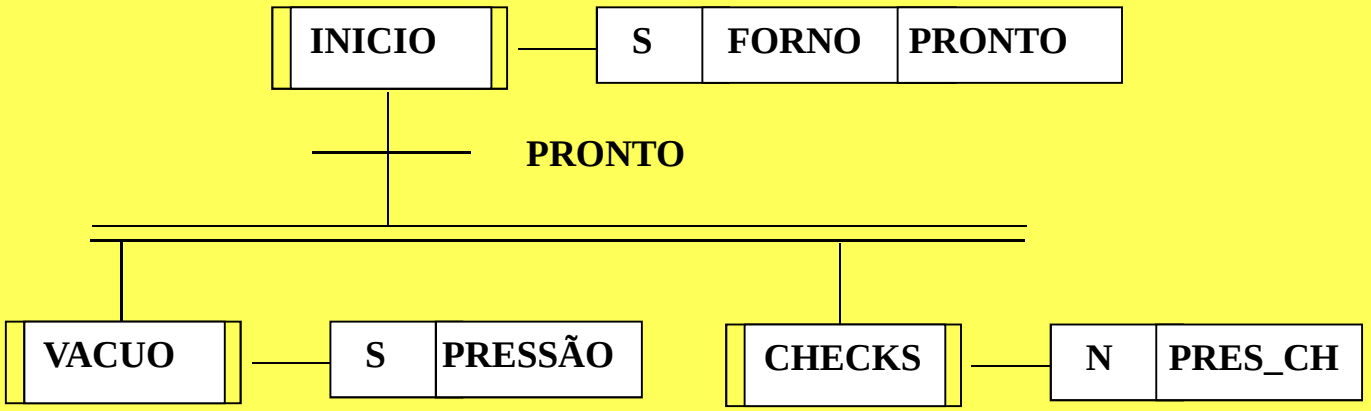
Obs. A chave retentiva do simulador quando desligada deve desligar todas as lâmpadas do semáforo. Caso não tenha o simulador faça somente o programa em ladder.




Botão_liga
TEMP_L2_DN
TEMP_L1
Pré 6.000
Acc 0000
TEMP_L1_DN
TEMP_L3_DN
Botão_liga
TEMP_L2
Pré 3.000
Acc 0000
TEMP_L2_EN
TEMP_L2_DN
Botão_liga
TEMP_L3
Pré 9.000
Acc 0000
TEMP_L3_EN
TEMP_L2_DN
TEMP_L1_TT
L1
TEMP_L2_TT
L2
TEMP_L3_TT
L3

Linguagem SFC

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



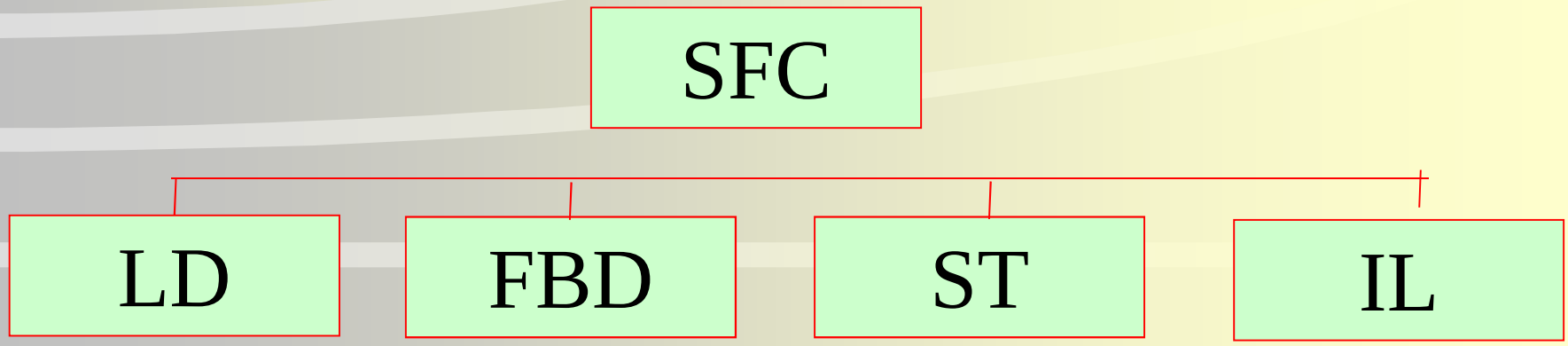
A linguagem SFC emprega passos (ou estados) e transições para ilustrar a execução de suas operações ou ações. Normalmente a Linguagem SFC é conveniente quando o processo possui:

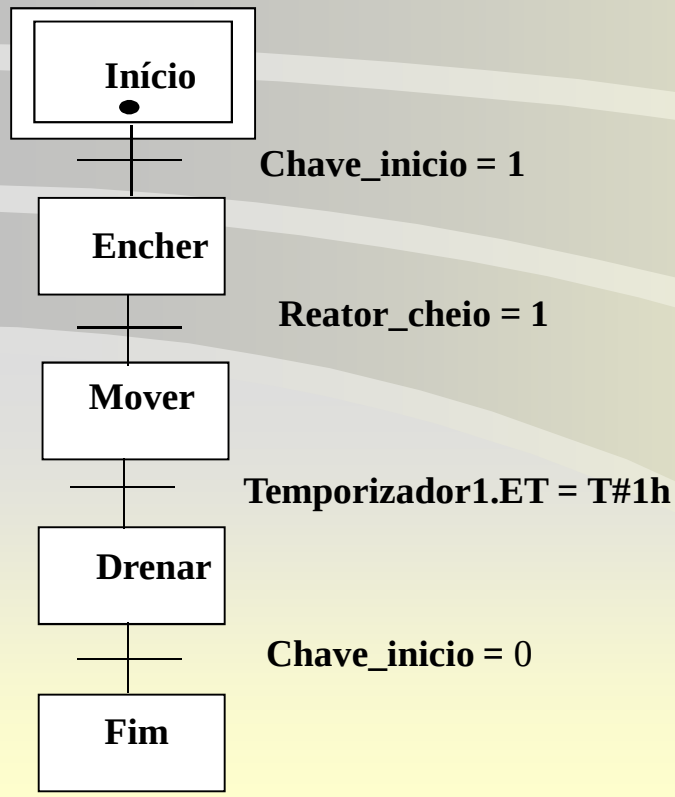
a - Gerenciamento de alto nível de múltiplas operações;
b - Sequências repetitivas de operações;
c - Processos por Batelada;
d - Controle tipo *Motion* (controle de movimento);
f - Operações do tipo máquina de estados.

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

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

No SFC um passo está ou não ativo. Quando um passo está ativo, um conjunto associado de instruções (definido como ações) é executado repetidamente até o passo se tornar inativo. A decisão de alterar o status de um passo (ou seja, tornar um passo ativo ou inativo, ou vice-versa) é determinada pela transição, que é o elemento imediatamente seguido do passo. A transição é programada por meio de uma condição de transição, através de uma expressão booleana. Quando a expressão se torna verdadeira (VERDADEIRA/TRUE) o passo ativo é desativado e um próximo passo é assumido como ativo.





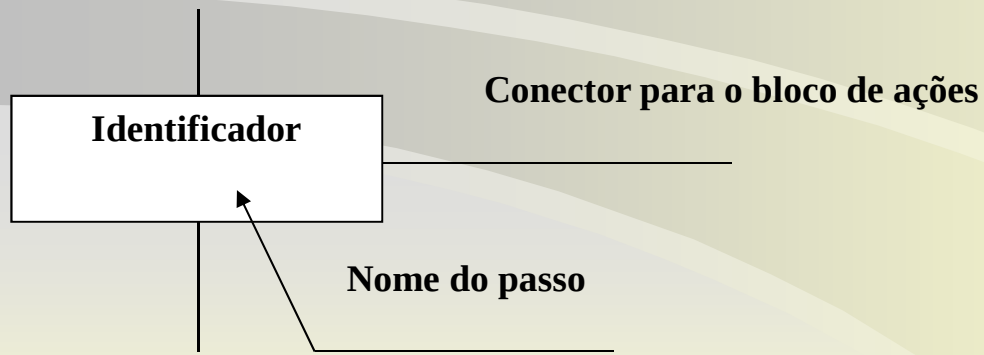
HIERARQUIA

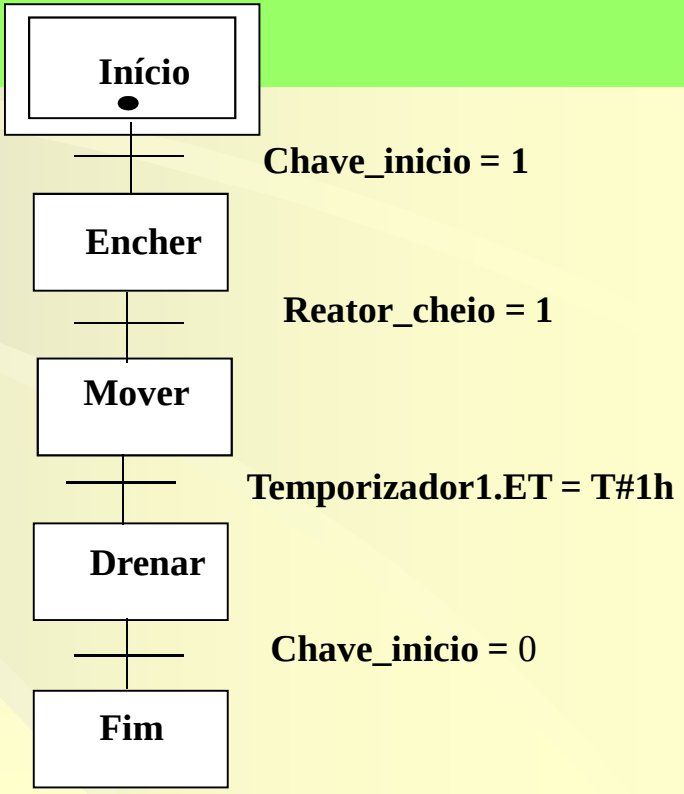
Cada ação é programada e encapsulada em LD, FBD, ST ou IL.

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

Variável FLAG, que indica que o passo está em atividade. Essa variável tem a nomenclatura (nome do passo).X.

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

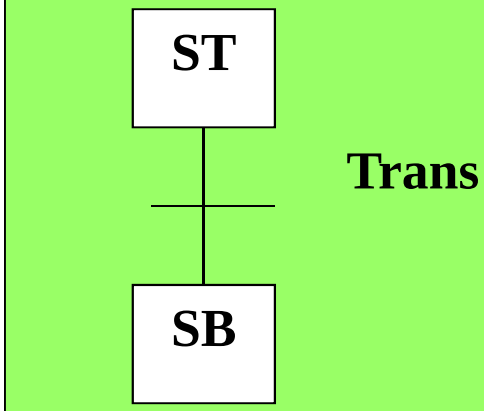




Regras de evolução do SFC e do GRAFCET

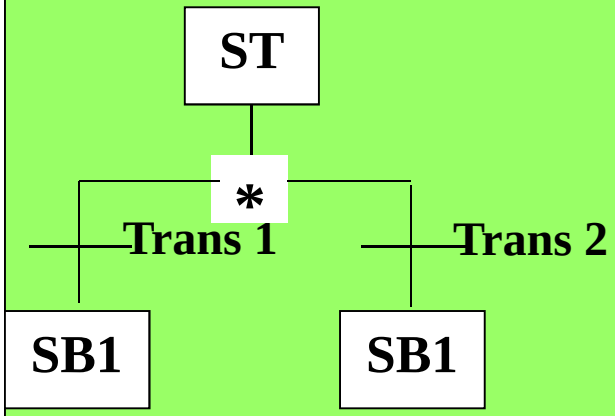
A transição é uma barreira entre os passos. Ela impede o fluxo de execução até que sua condição seja satisfeita e até que a própria transição esteja habilitada. Uma transição está habilitada se todos os estágios ou passos anteriores estão ativos.

As condições de uma transição podem ser expressões lógicas, temporais, aritméticas, etc. e são denominadas de receptividade da transição. Na ocorrência de uma transição, ocorre a ativação de todos os estágios ou passos imediatamente posteriores e que estão conectados a esse transição. Numa transição também são desativados todos os estágios imediatamente precedentes e a ela conectados.



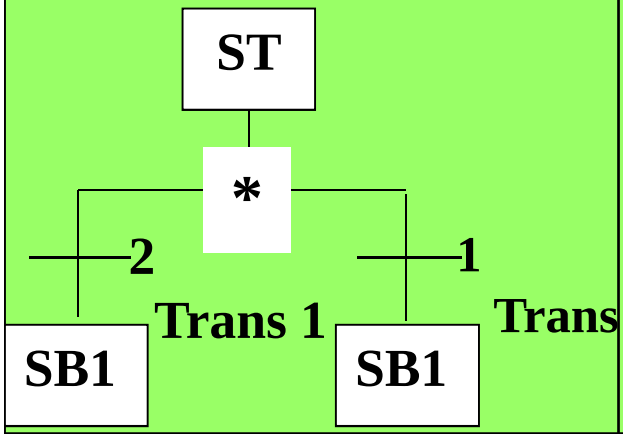
Sequência simples

- ST é desativado, tão logo, Trans=True (transição=verdadeira).
- SB torna-se ativo assim que ST é desabilitado ou tornado inativo.



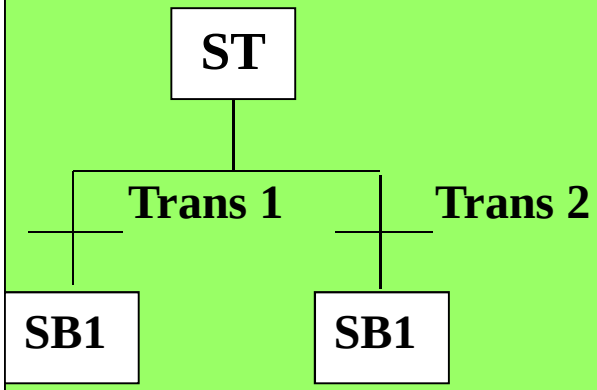
Caminho divergente

- Quando ST está ativo, as transições 1 e 2 são avaliadas segundo a ordem da direita para a esquerda.
- Tão logo uma transição seja TRUE (Verdadeira), ST é colocado inativo e a SB subsequente é colocada ativa.



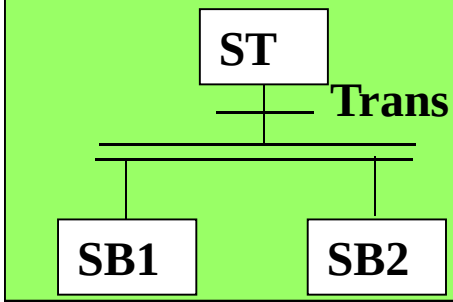
Caminho divergente com prioridade definida pelo usuário

- O usuário define a prioridade de precedência. O número menor tem a prioridade mais alta.



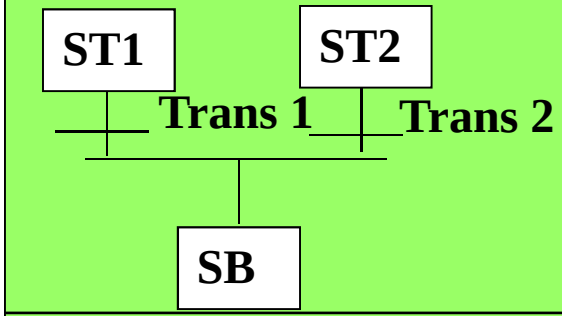
Caminho divergente sob o controle do usuário

- O usuário deve garantir que as duas ou mais transições sejam mutuamente exclusivas.
- Obs.: Caso “Trans 1” e “Trans 2” venham a se tornar TRUE ao mesmo tempo, fica caracterizado o equivalente em Redes de Petri denominado “conflito confusão”.



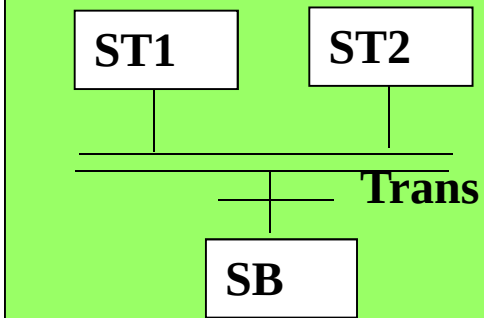
Sequência simultânea

- Quando a “Trans” é TRUE, todos os passos subsequentes tornam-se ativos simultaneamente.



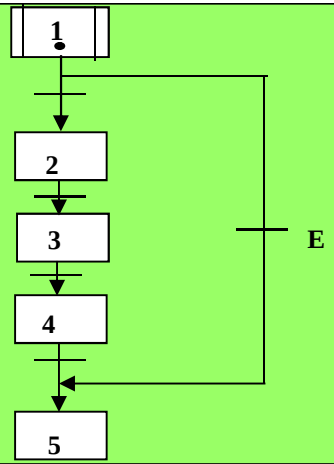
Convergência de sequência

- Quando um passo STn está ativo e sua “Trans n” sucessiva torna-se TRUE, o passo STn torna-se inativo e o passo SB é ativado.

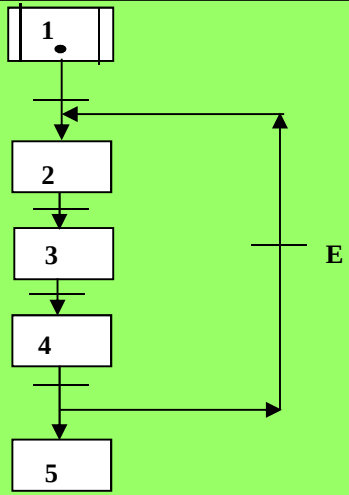


Convergência de sequência simultânea

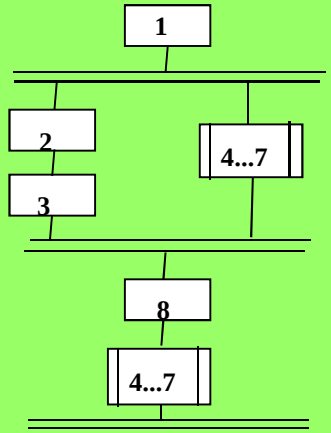
- Quando todos os passos STn estão ativos e todas as “Trans n” correspondentes tornam-se TRUE, os passos STn são desativados e o passo SB ativado.



Jump Condicional



Repetição Condicional



Sequência Repetitiva

Ações em SFC

A linguagem SFC permite representar uma série de ações básicas. A estrutura típica é: qualificador da ação, ação e indicador de variável

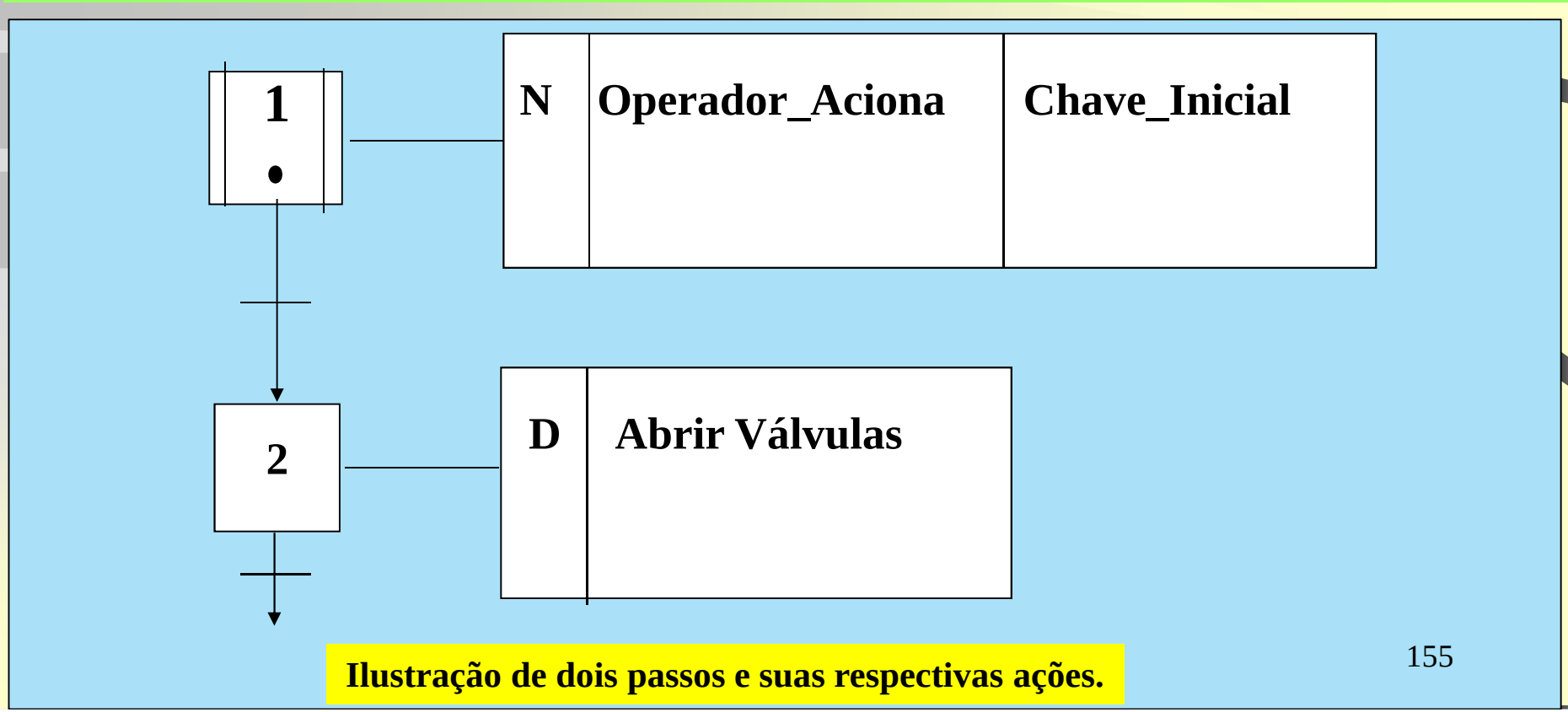
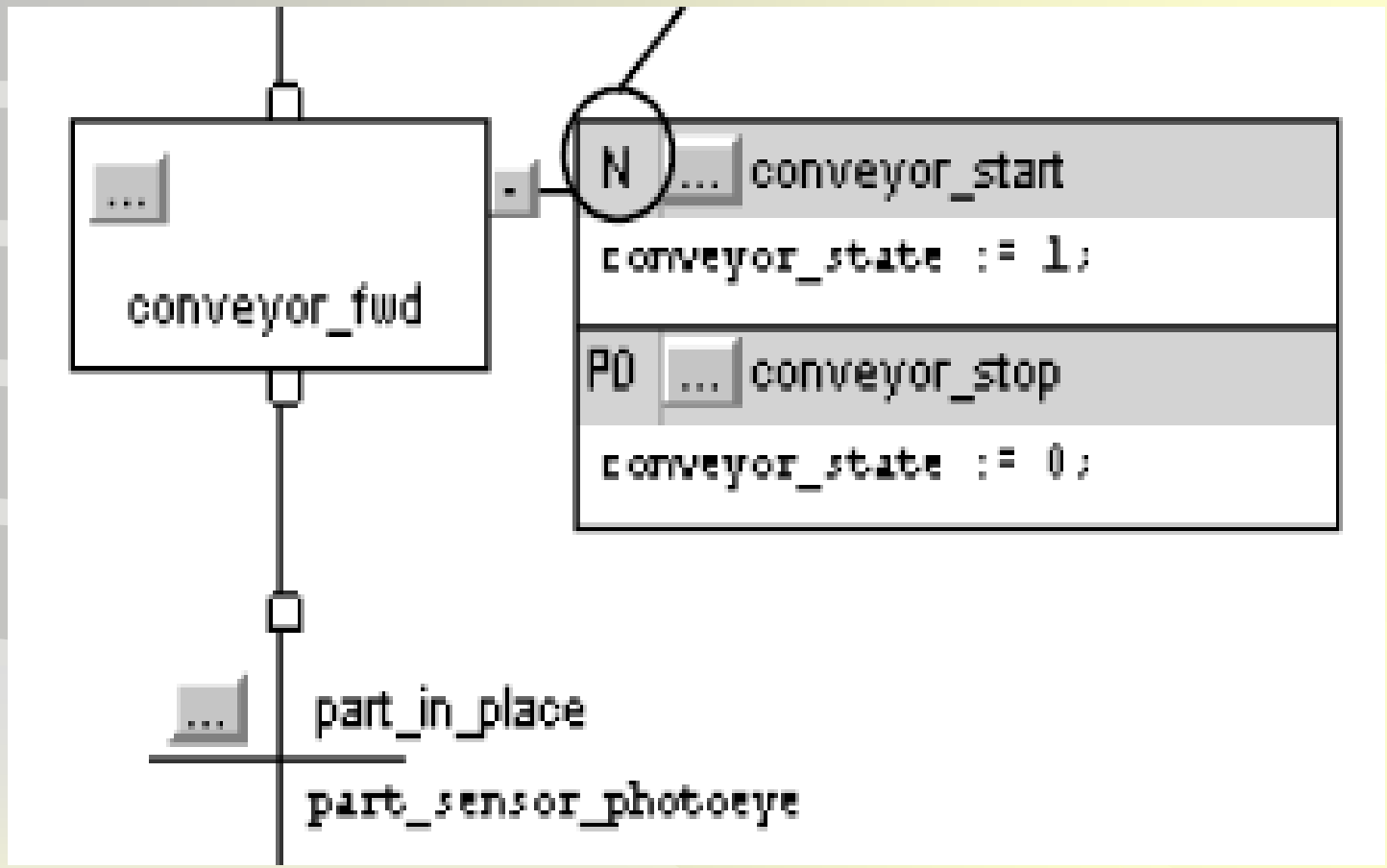


Ilustração de dois passos e suas respectivas ações.

Elementos estruturais do SFC

Passo



Ações

Transição

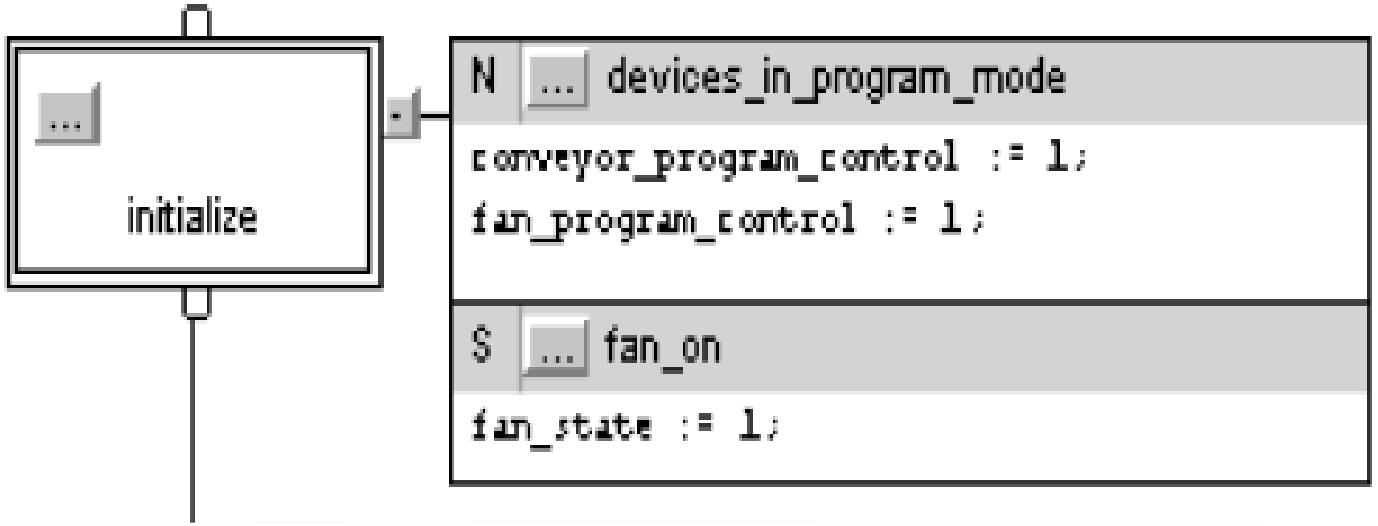
Qualificador	Descrição da ação
N	Ação simples. Executa a ação associada com o passo enquanto o mesmo está ativo. “Não memoriza”.
S	Set. Seta a ação ativa. “Memoriza”.
R	Reset. Reseta ou desativa a ação “setada” ou “memorizada”.
L	Ação por tempo limitado. Executa por um determinado tempo a ação.
D	Ação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

Principais ações na linguagem SFC.

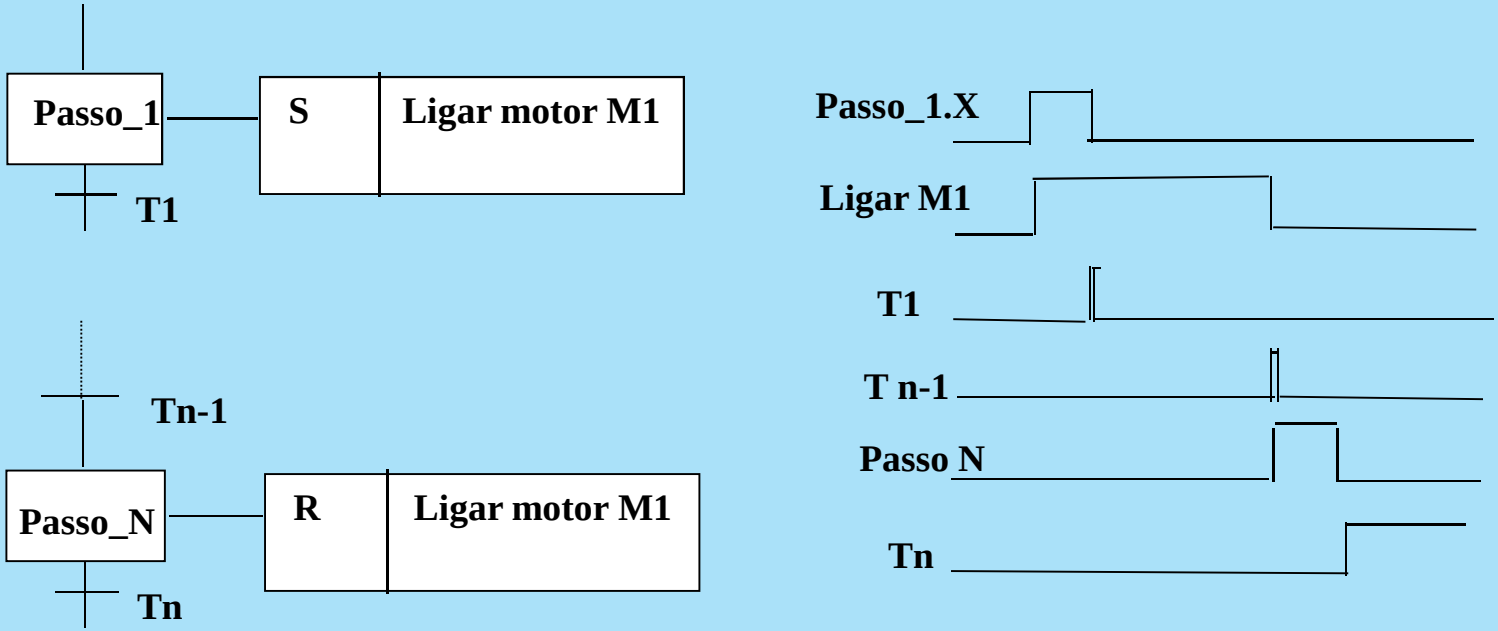
Elementos estruturais do SFC

Qualificadores das ações da linguagem Grafcet – resumo

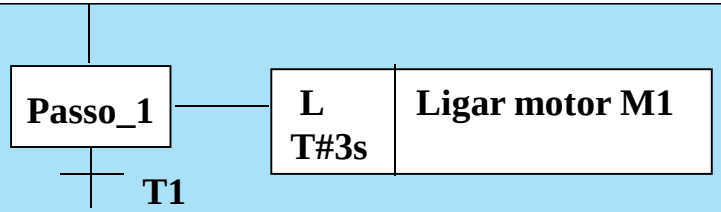
Letra L	Temporiza o tempo de execução da ação
Letra D	Retarda o tempo de entrada da ação
Letras S/R	Vincula a entrada ou saída da ação com passos do Grafcet
Letra N	Vincula a ação ao próprio passo em atividade

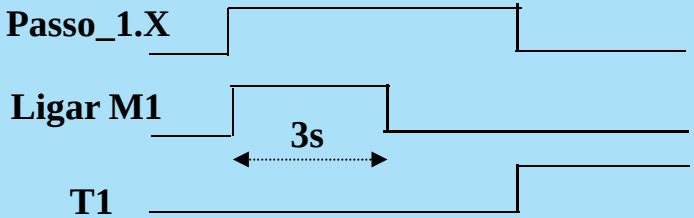


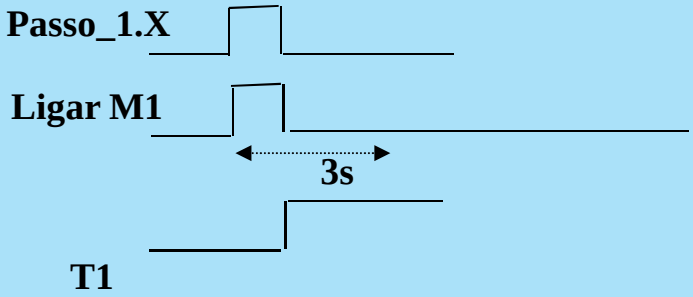
Exemplo de Ação Simples (N).



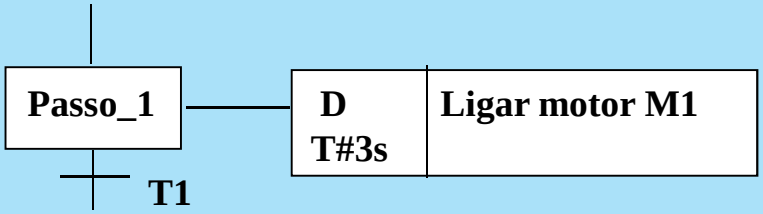
Exemplo das ações: Ação Set (S) e Ação Reset (R).

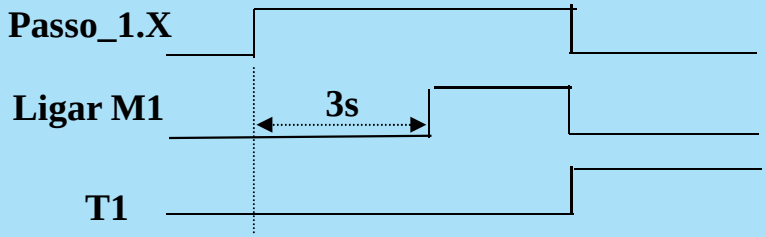


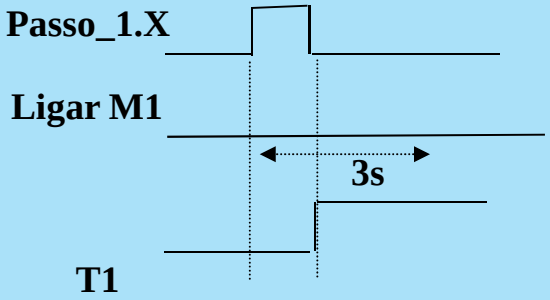




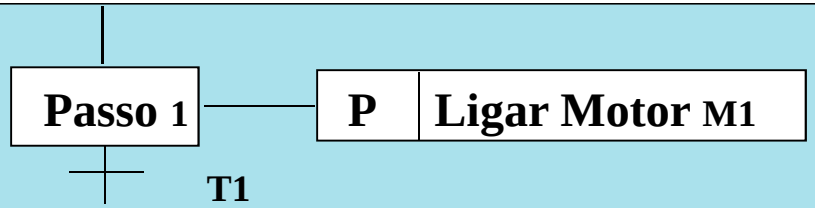
Exemplo da Ação Tempo Limitado (L).



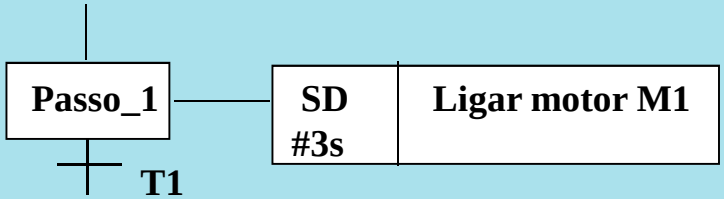


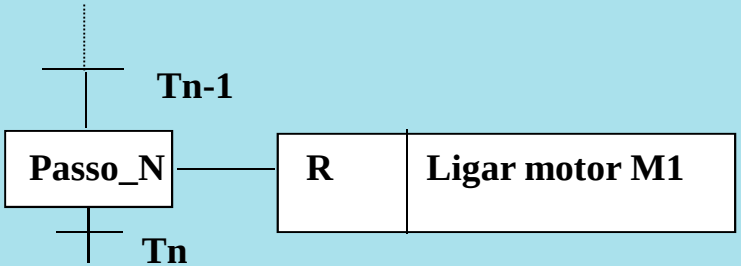


Exemplo da Ação Tempo Retardado (D)

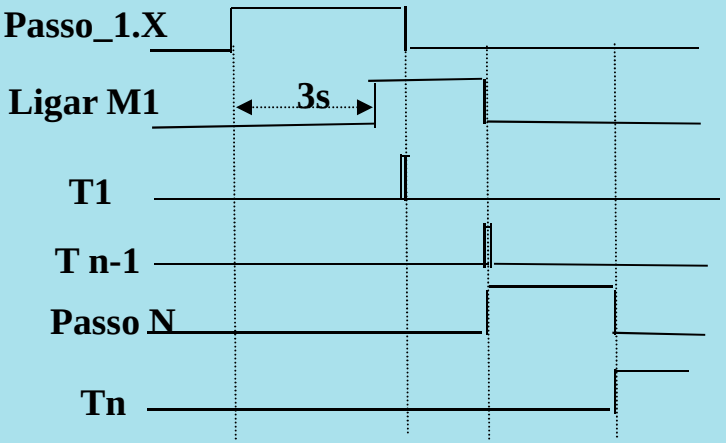


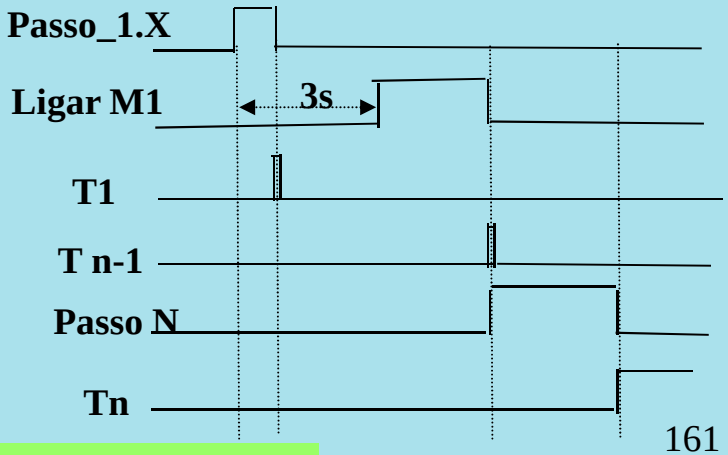
Exemplo da Ação Pulsada (P)



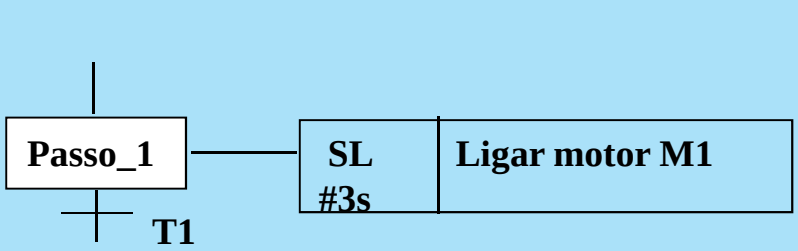


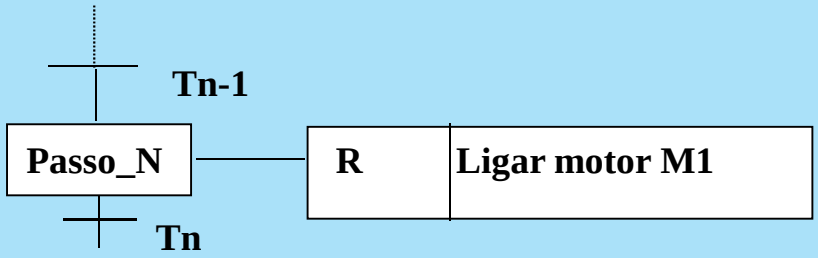
Quando o passo 1 for ativado, a ação é memorizada e somente começará depois do tempo especificado. A Ação continuará até ser referendada no passo N com o Reset. Mesmo que o passo 1 seja desabilitado antes do início da ação a mesma ocorrerá caso o passo relativo ao Reset não esteja habilitado.

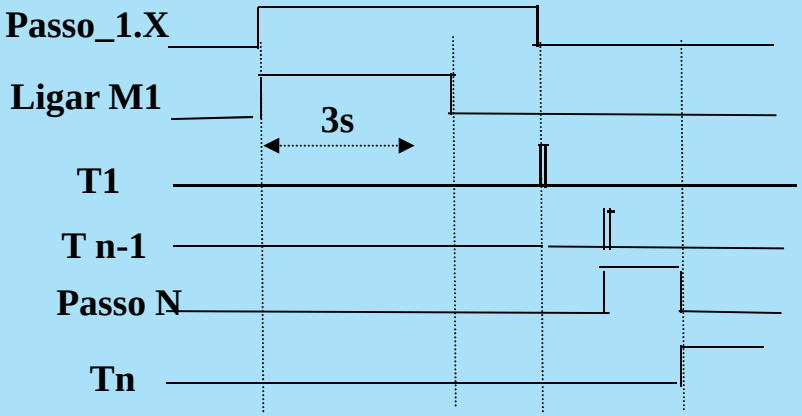


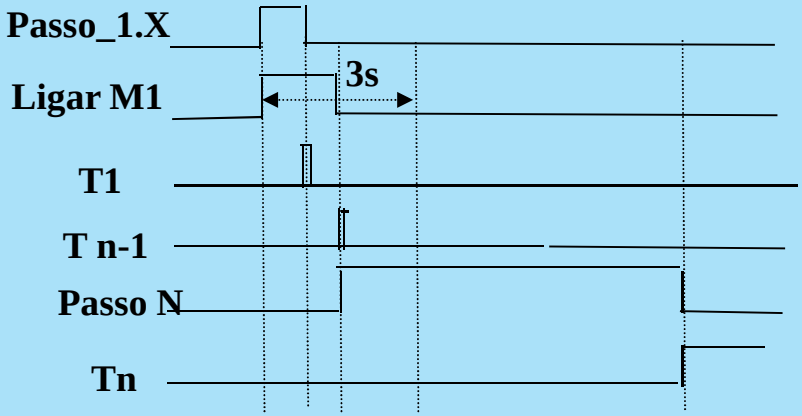


Exemplo da Ação de Entrada com Tempo Prefixado (SD)





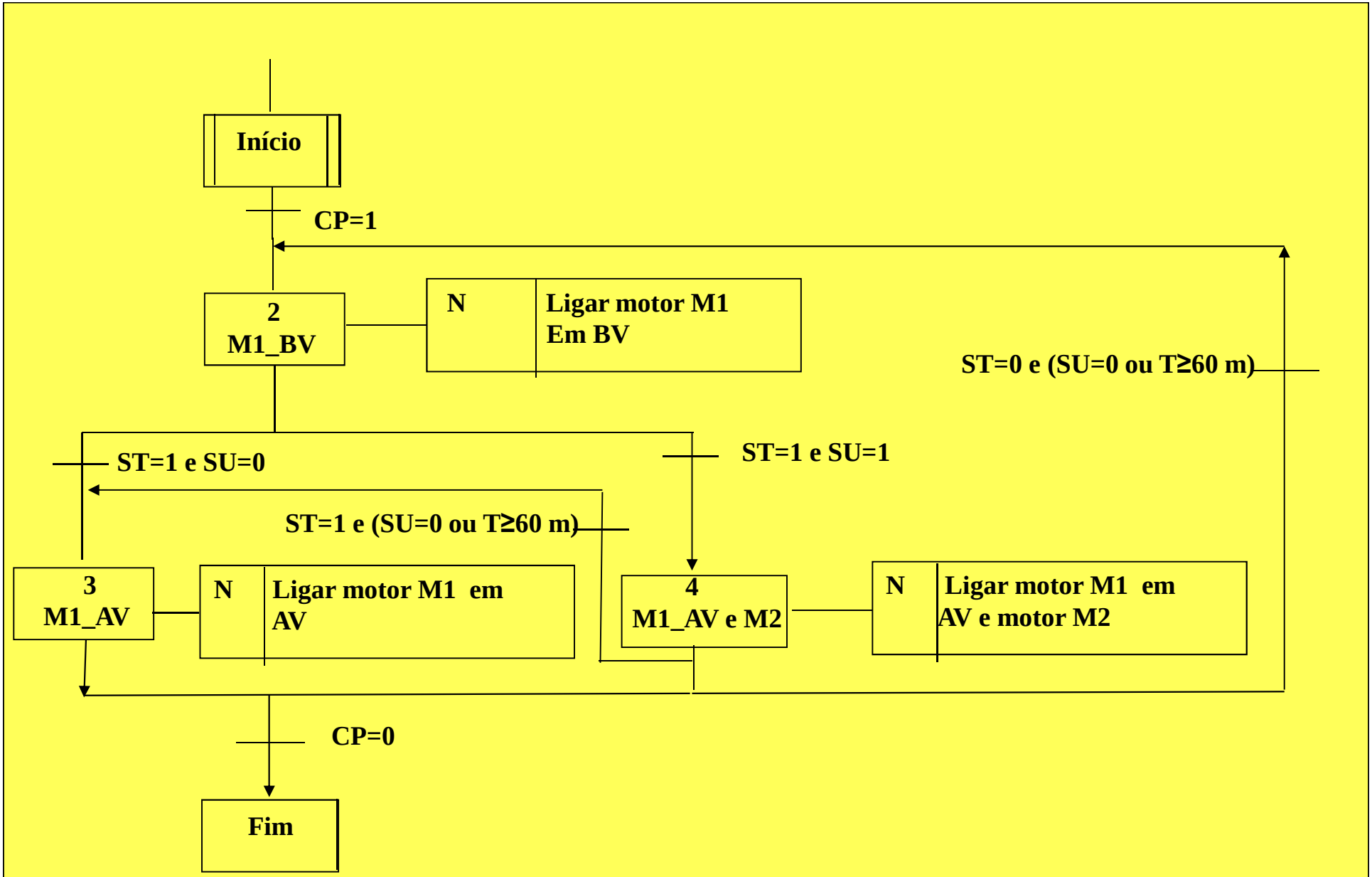




Exemplo da Ação com Tempo Limitado (SL).

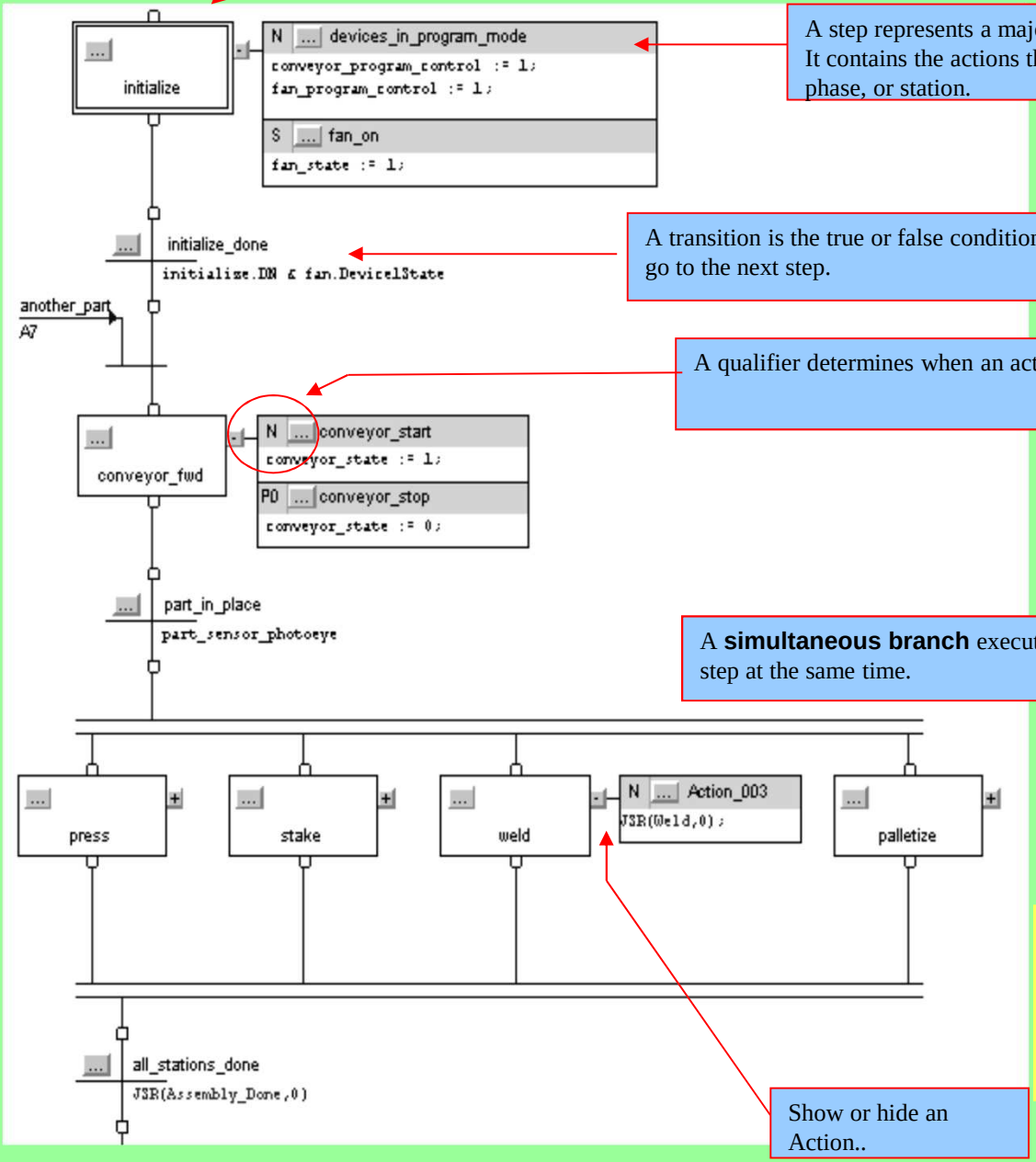
EXEMPLO:

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



Programa em SFC para executar o exemplo exposto

A step represents a major function of your process. It contains the actions that occur at a particular time, phase, or station.



A step represents a major function of your process. It contains the actions that occur at a particular time, phase, or station.

A transition is the true or false condition that tells the SFC when to go to the next step.

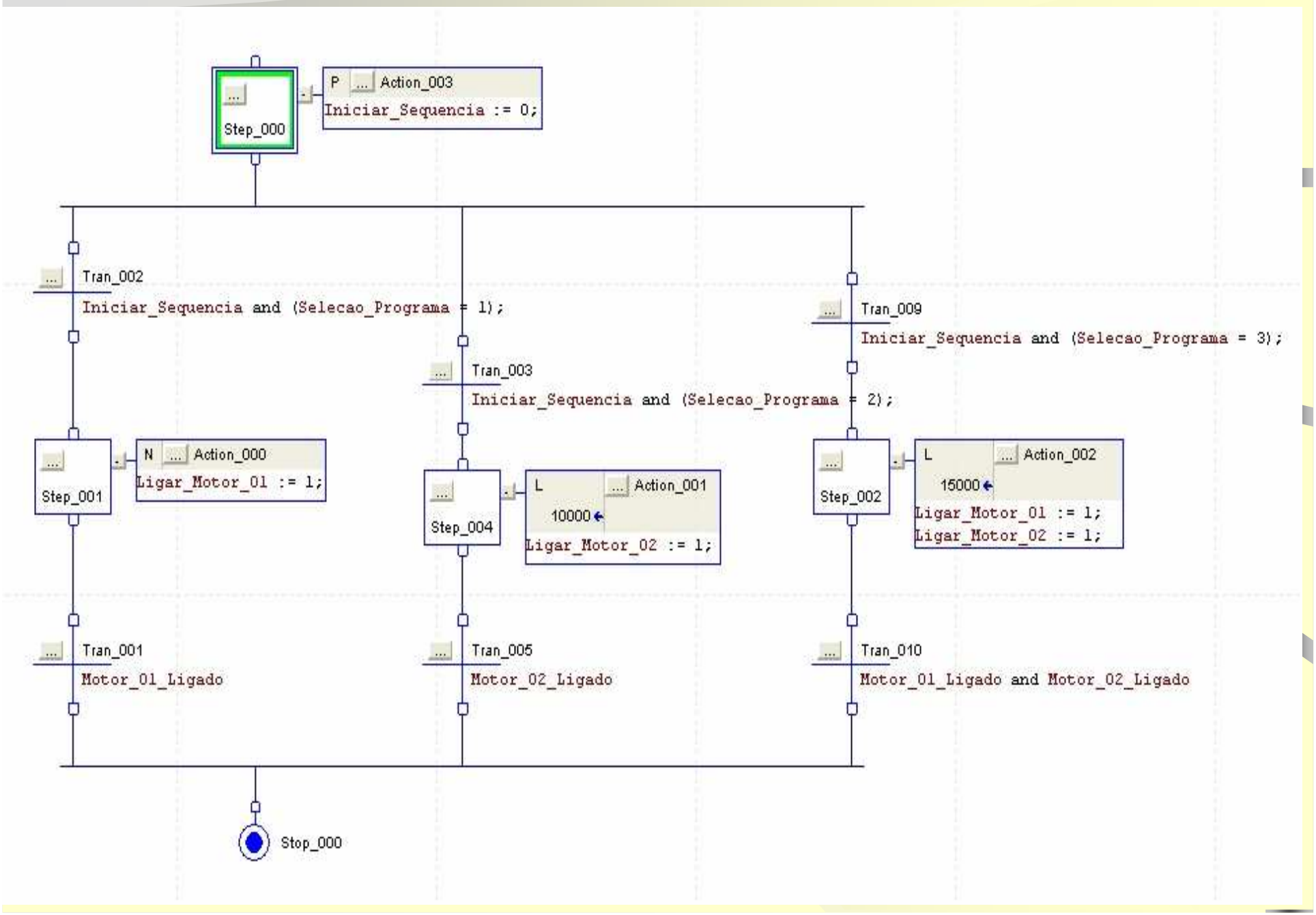
A qualifier determines when an action starts and stops.

A **simultaneous branch** executes more than 1 step at the same time.

Show or hide an Action..

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

Exemplo – Sequenciamento 1



Exemplo – Sequenciamento

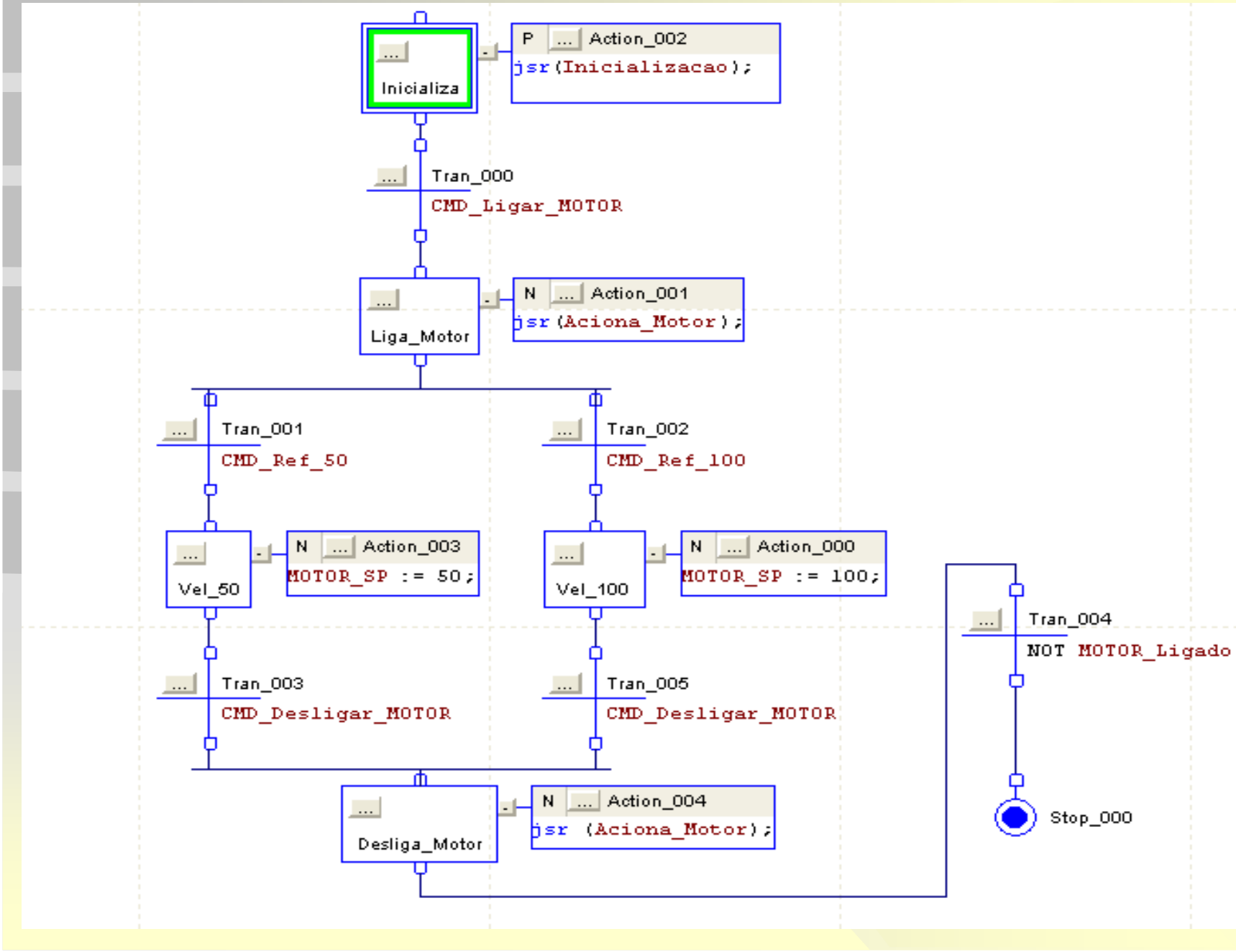


Diagrama de Blocos de Funções “Function Blocks Diagram (FBD)”:

Um conceito importante para a melhor compreensão e utilização das linguagens do padrão IEC é a sigla “POU” “Program Organization Unit”.

Uma POU é a menor e independente unidade de software de um programa e corresponde a um bloco que pode chamar outra POU com ou sem parâmetros. Uma POU pode ser programada em qualquer uma das linguagens existentes

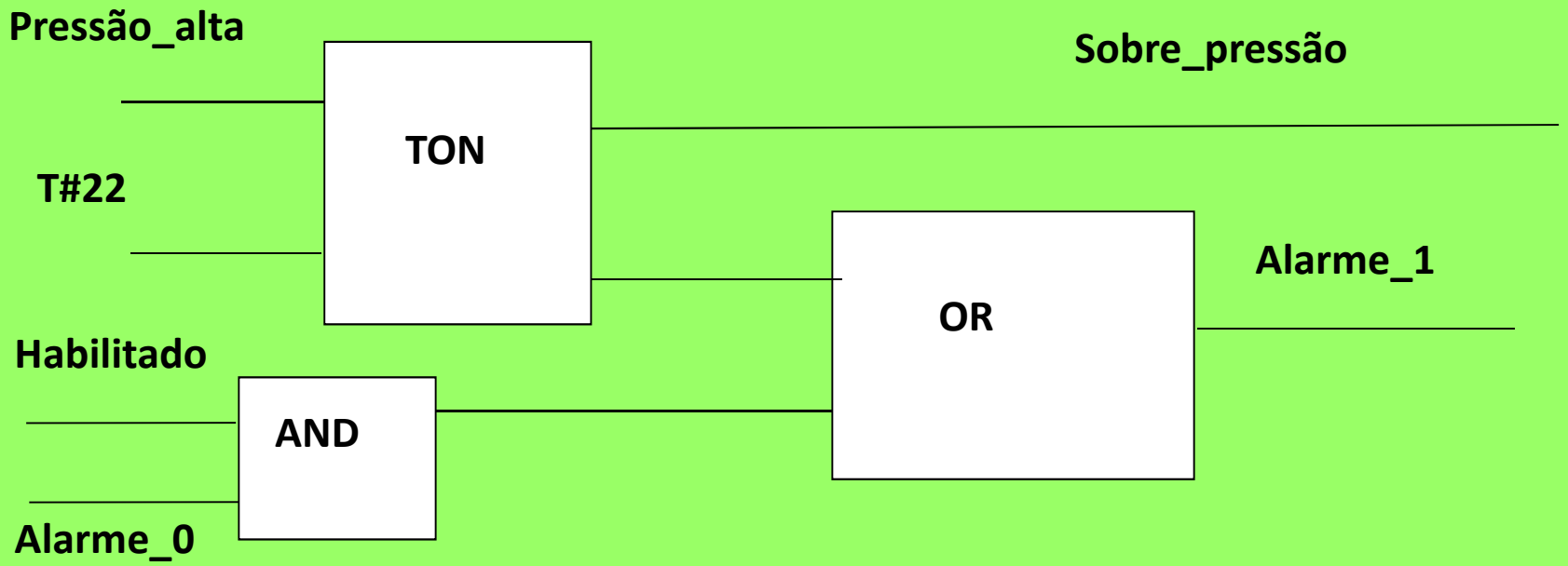
Existem três tipos de POUs:

A – “FUNCTION” FUN: É UMA POU QUE PODE SER ATRIBUÍDA COM PARÂMETROS NÃO ESTÁTICOS (SEM MEMÓRIA) E QUE QUANDO CHAMADAS COM OS MESMOS VALORES DE ENTRADA RETORNAM OS MESMOS RESULTADOS COMO FUNÇÕES.

B – “FUNTION BLOCK” FB: É UMA POU QUE PODE SER ATRIBUÍDA COM PARÂMETROS E POSSUI VARIÁVEIS BÁSICAS (COM MEMÓRIA). POR EXEMPLO: UM CONTADOR OU UM TEMPORIZADOR É UMA “FUNCTION BLOCK” FB.

C – “PROGRMAM” (PROG) É A POU QUE REPRESENTA O ‘MAIN PROGRAM” OU SEJA O PROGRAMA PRINCIPAL.

A representação de uma POU na linguagem FBD ou mesmo na linguagem LD incluem partes como nas linguagens textuais como: parte principal e parte final da POU, parte da declaração e parte do código



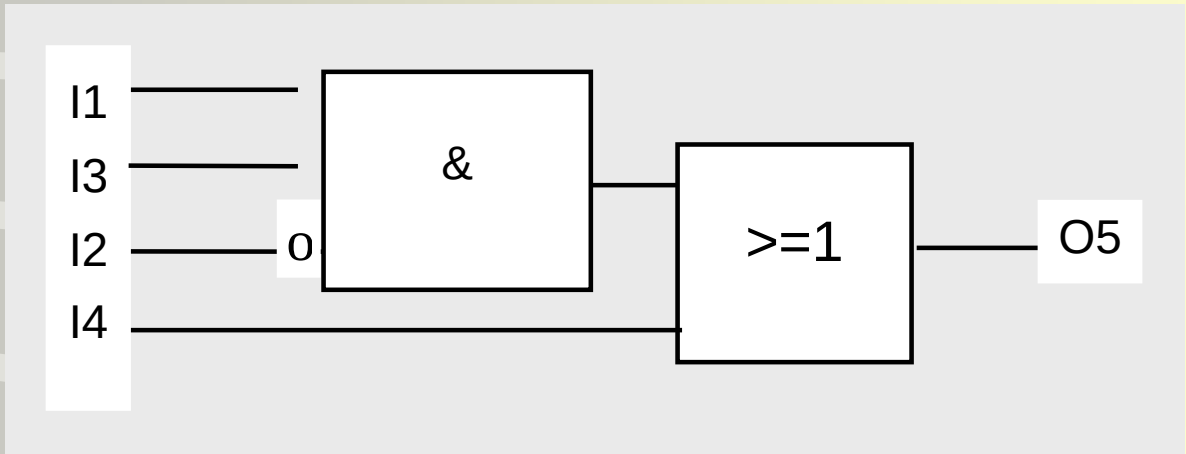
Exemplo de um trecho de programa escrito em linguagem FBD.

Function Block Diagram (FBD)

Utilizar Linguagem bloco de funções quando o processo possui:

- **Controle de drives** e **processos contínuos**;
- **Loop de controle**.

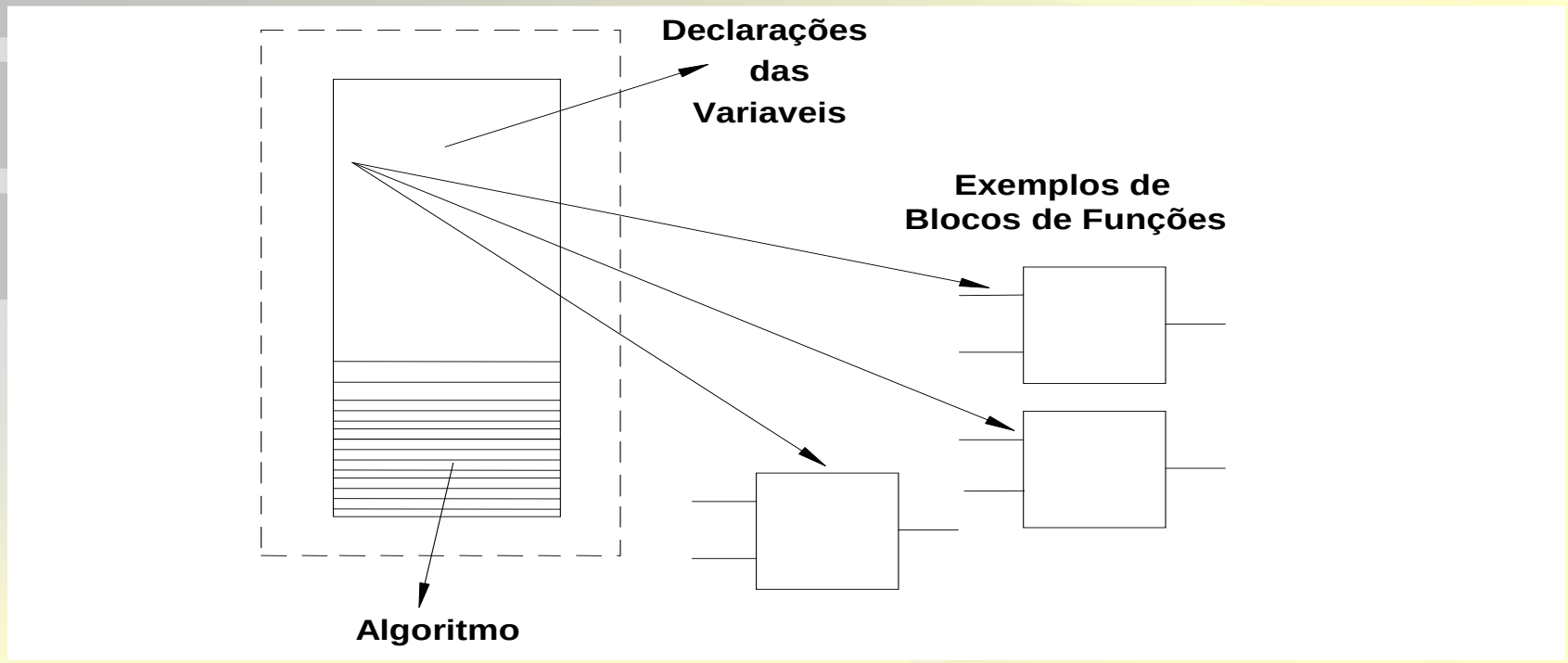
Linguagem de programação que permite especificar algoritmos ou um conjunto de ações a serem aplicados a um conjunto de dados de entrada para um conjunto de dados de saída.



Exemplo de Function Block Diagram

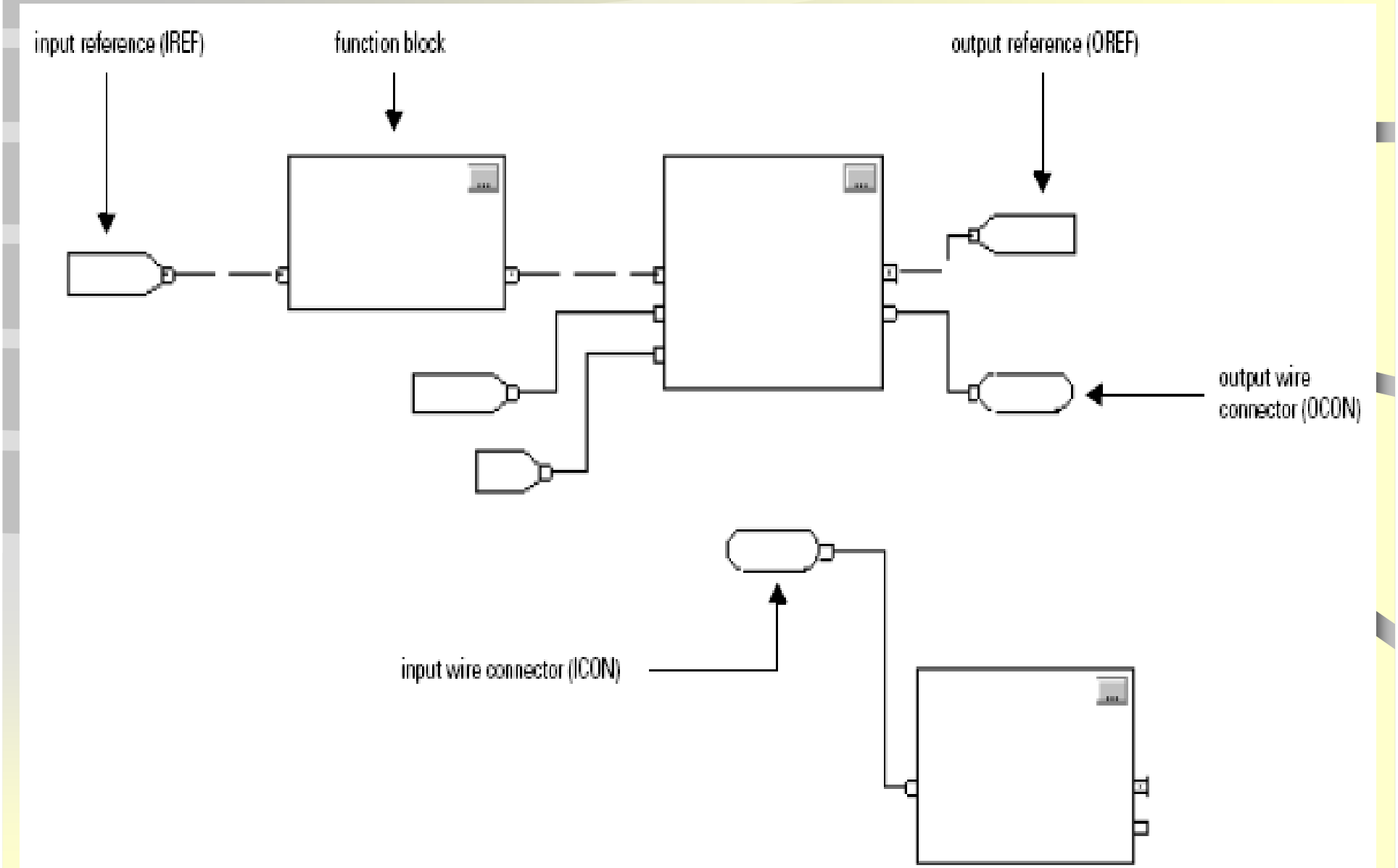
- Uma especificação de estrutura de dados constituída de parâmetros de entrada, variáveis internas e parâmetros de saída usando declarações textuais.

- Um algoritmo que poderá ser expresso usando qualquer linguagem de programação definida na IEC 61131-3



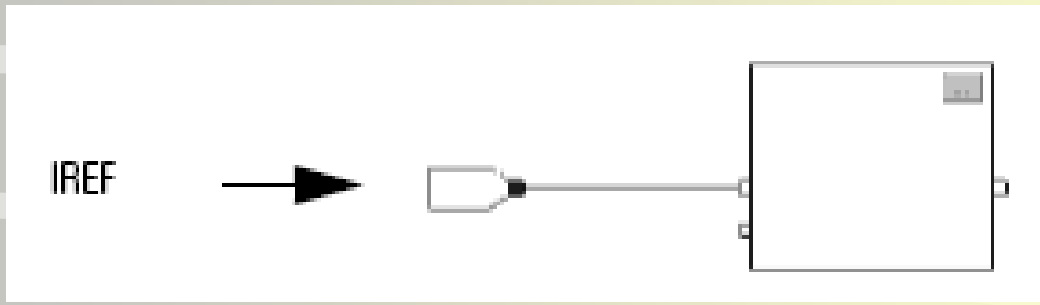
Estrutura de um bloco de funções

Visão geral de um diagrama de Blocos

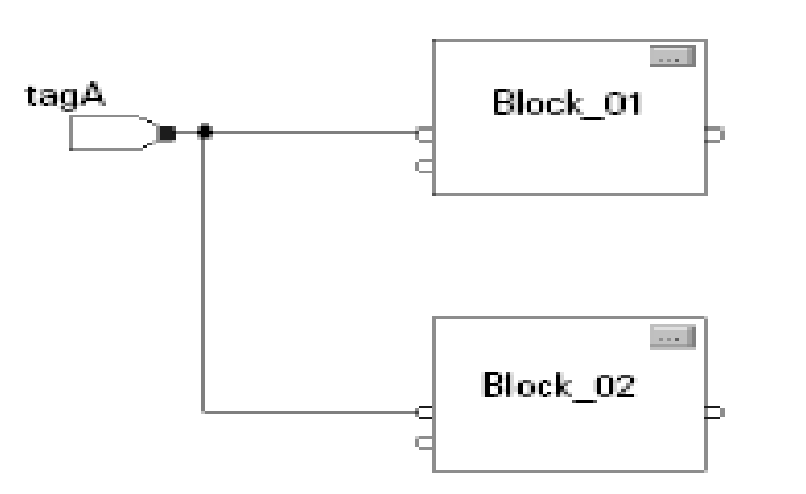


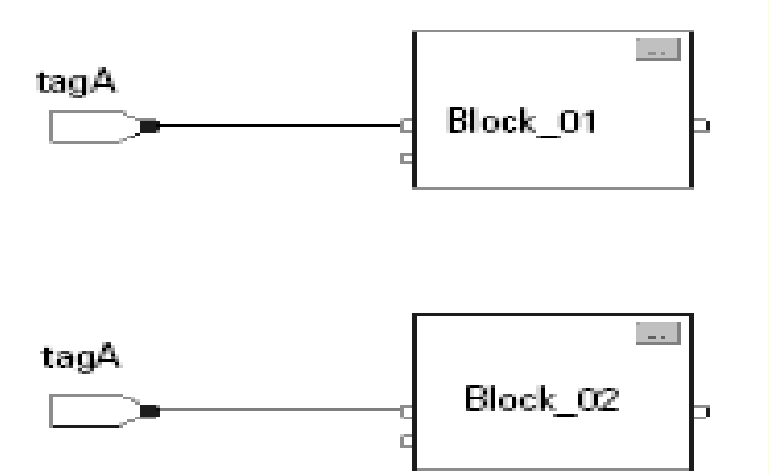
Atualização e utilização dos dados de entrada pelo Bloco de Função

- O controlador atualiza os valores de todos os IREF´s (referências de entrada) no começo de cada scan (varredura).



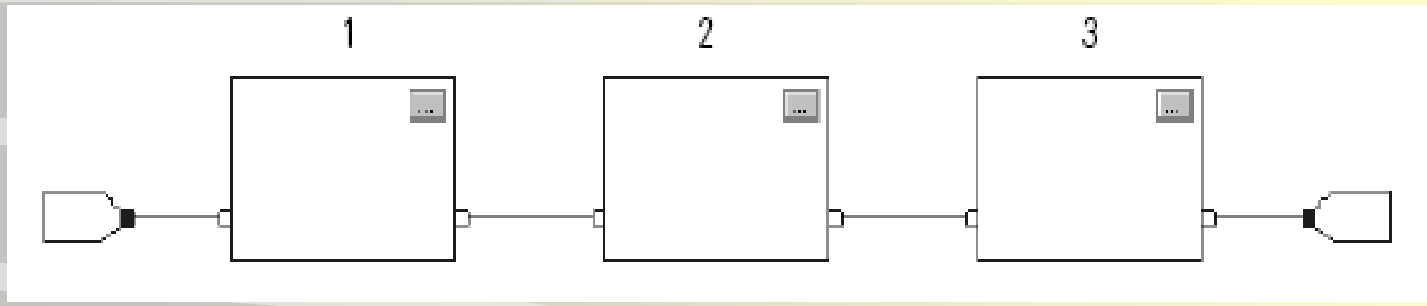
- O mesmo valor atualizado do tagA é utilizado na execução de cada um dos dois blocos.



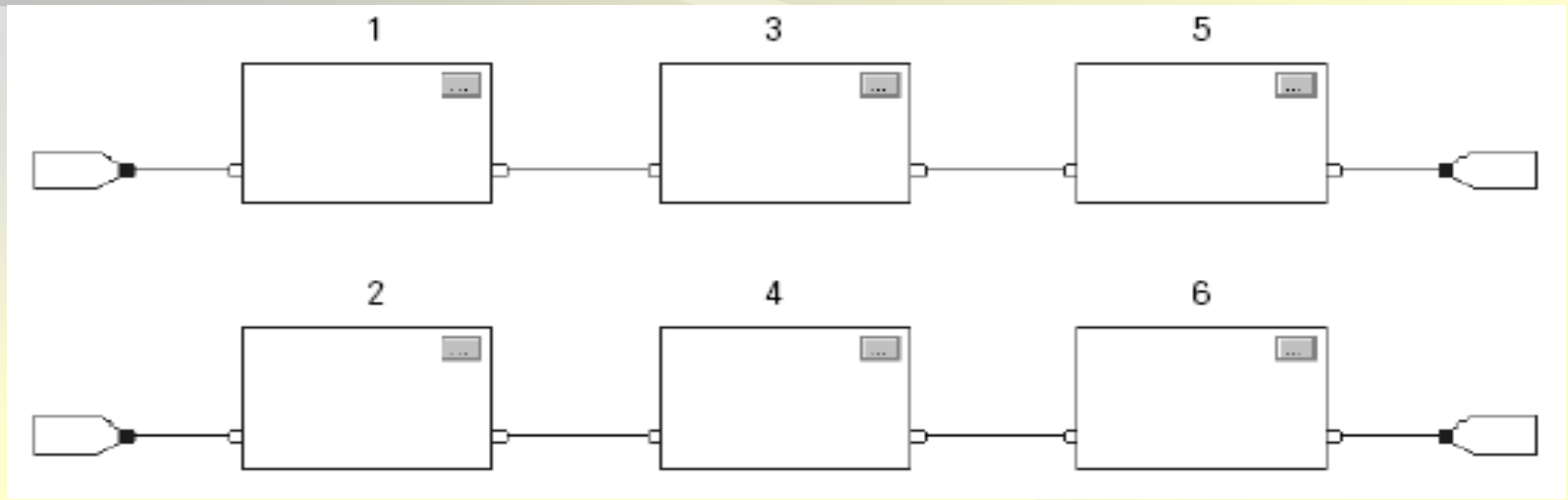


Ordem de execução dos Blocos de funções

- A ordem de execução é da entrada para a saída no caso de blocos conectados sequencialmente.

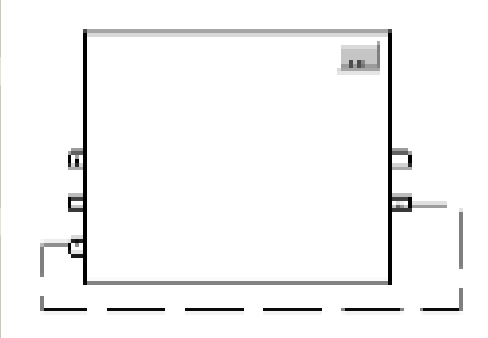


- Os conjuntos de blocos são executados das entradas para as saídas.

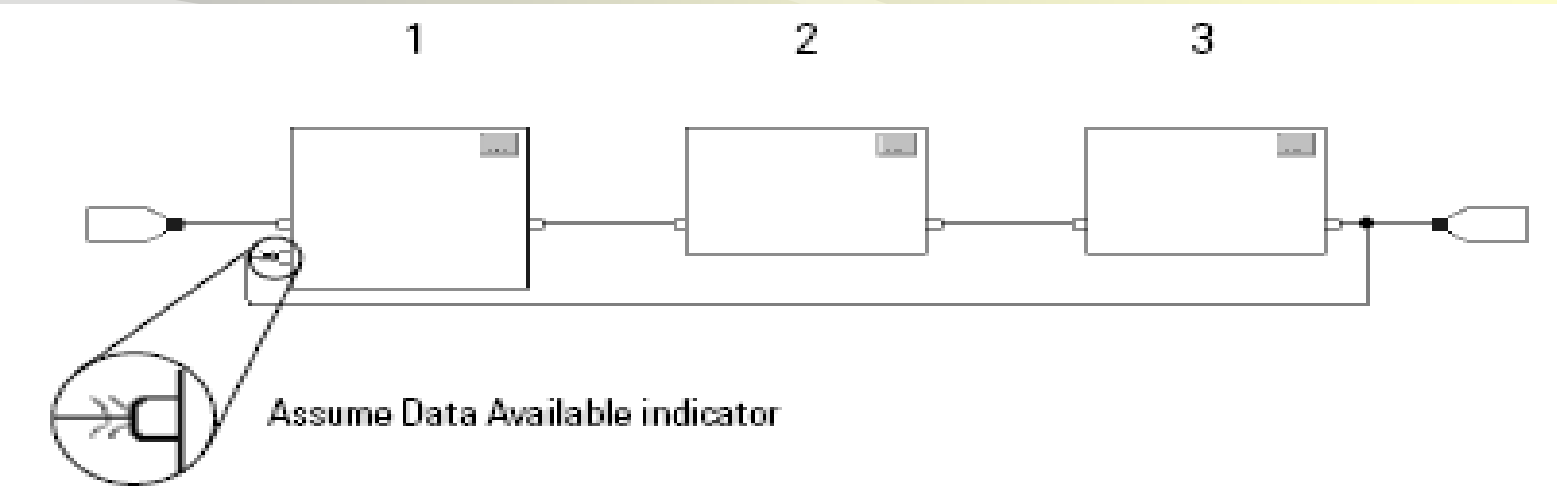


Ordem de execução dos Blocos de funções

- Execução de loop unitário (entrada e saída do mesmo bloco).



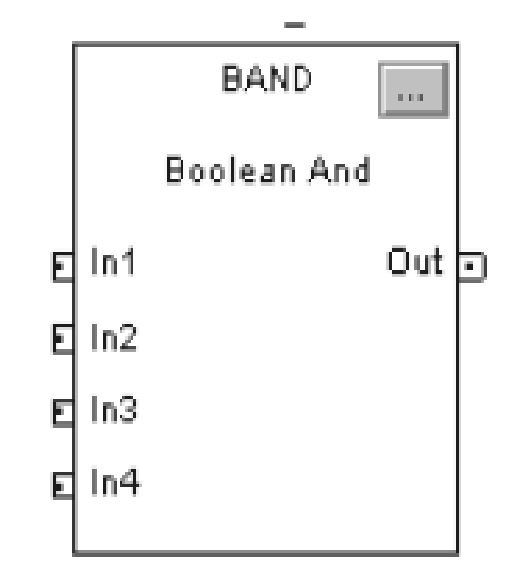
- Execução de loop composto (entrada e saída de blocos diferentes). É utilizado o dado de saída do Bloco 3 gerado no scan anterior.

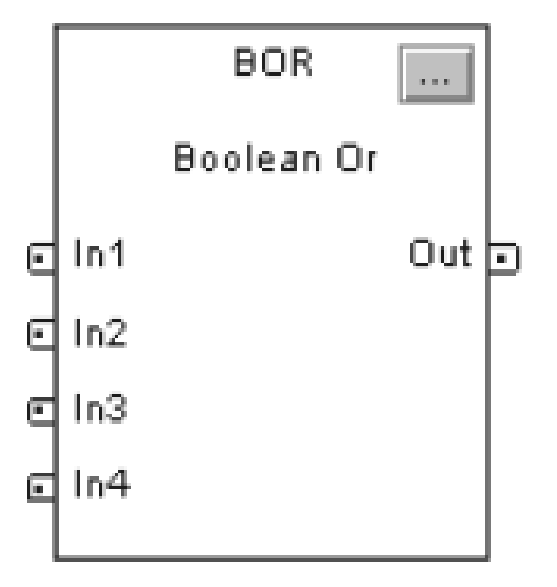


Lógica de relé ou instrução de Bit

BAND – E booleano (Boolean AND) e

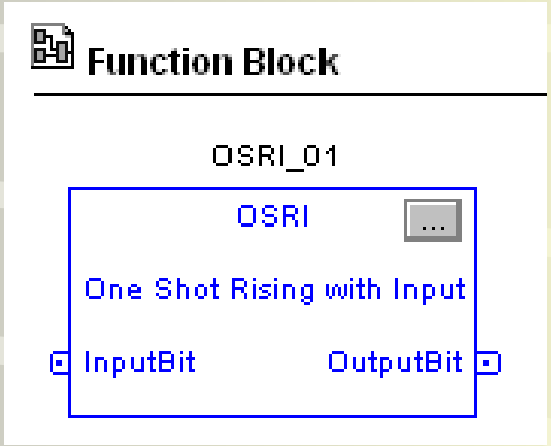
BOR – Ou booleano (Boolean OR)

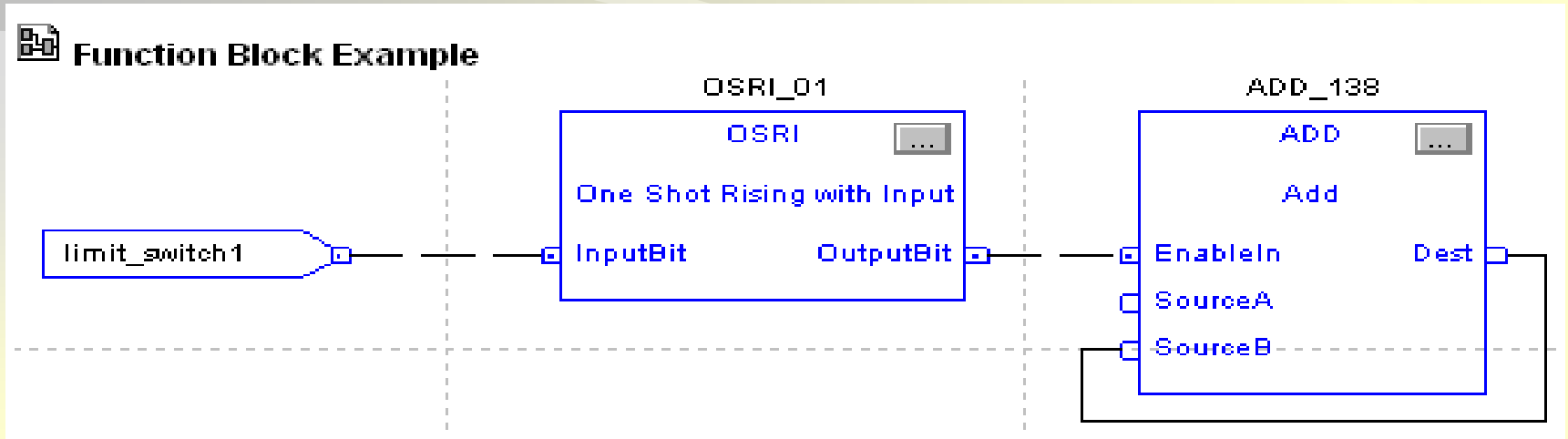




Lógica de relé ou instrução de Bit

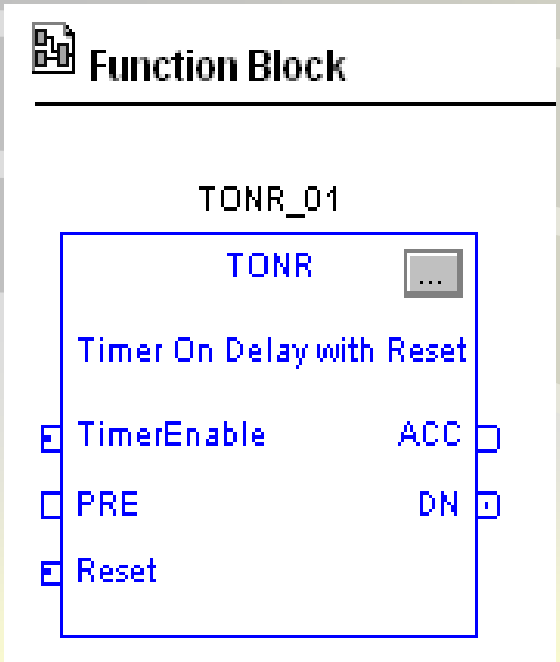
OSRI – Detector de borda de subida (One Shot Rise with Input)

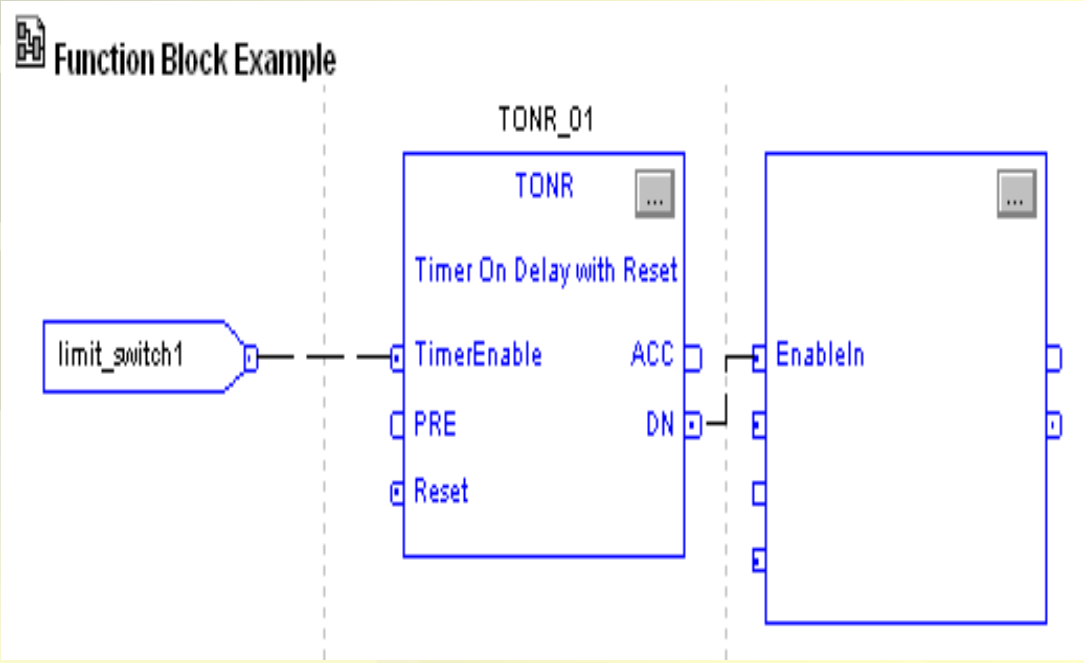




Temporização

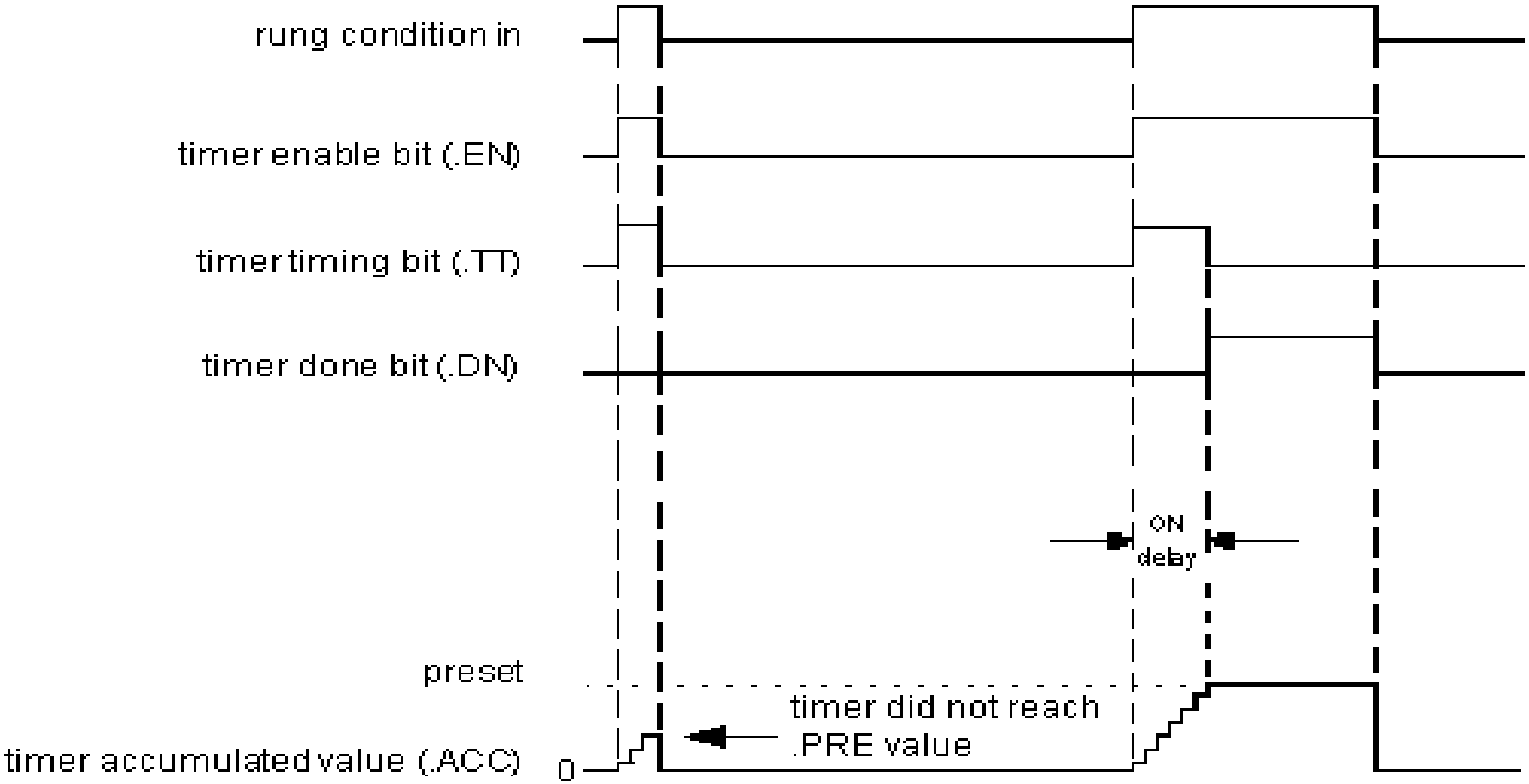
Instrução TONR - Temporizador crescente sem retenção à Energização com reset (Timer ON Delay with Reset)





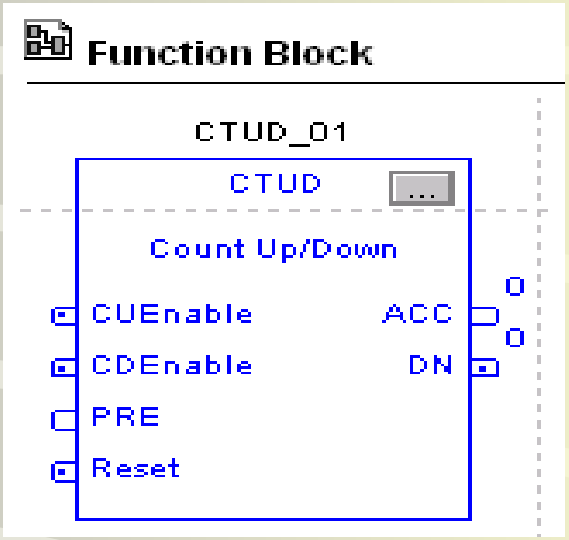
Temporizador crescente sem retenção à Energização (TON

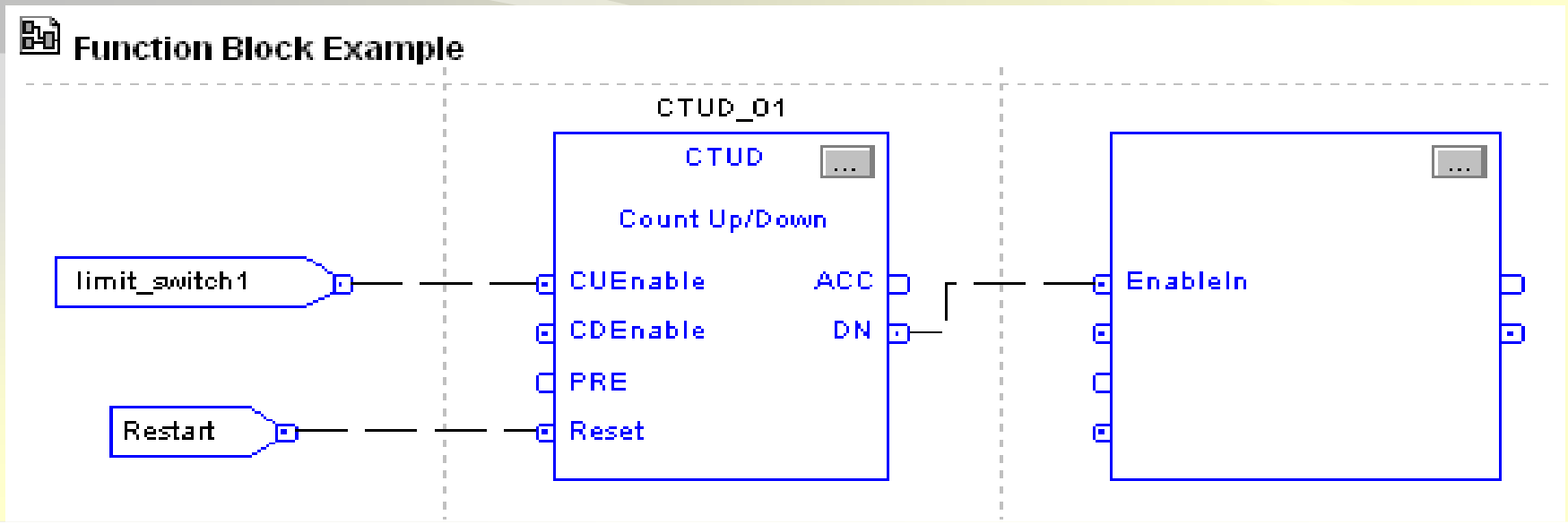
Carta de tempo



Contagem

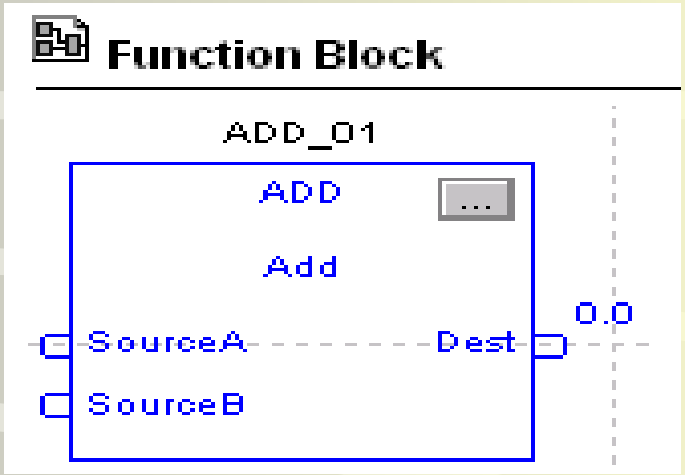
Instrução CTUD - Contador crescente/decrescente (CounT Up and Down)

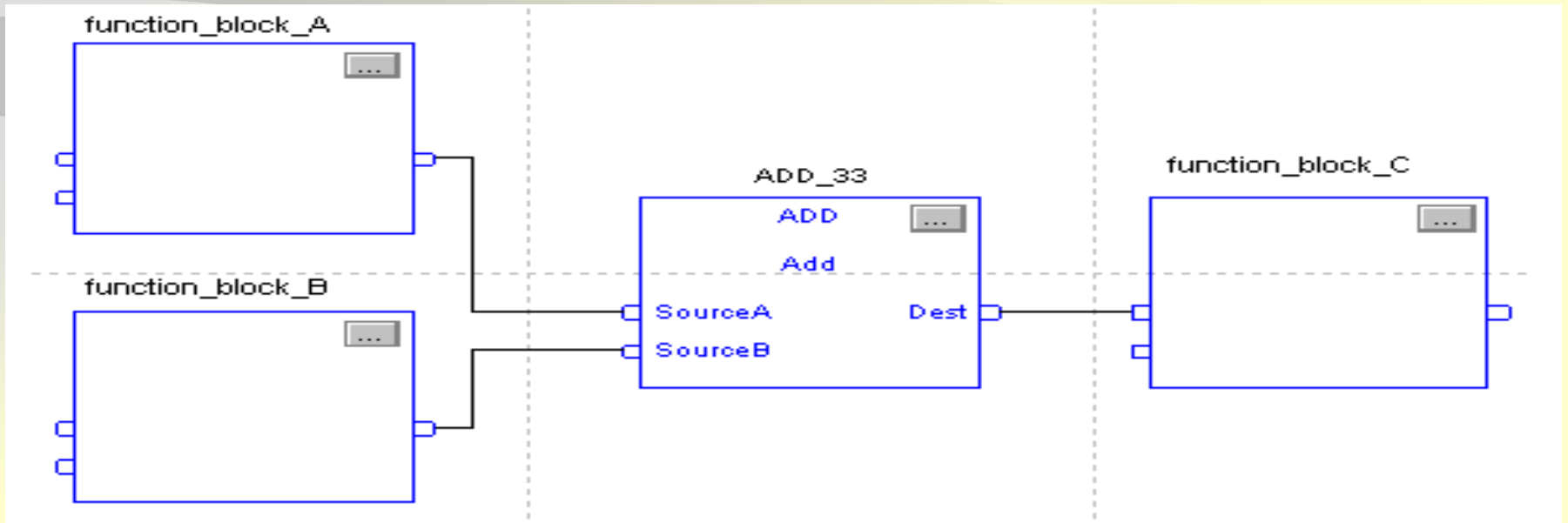




Aritméticas

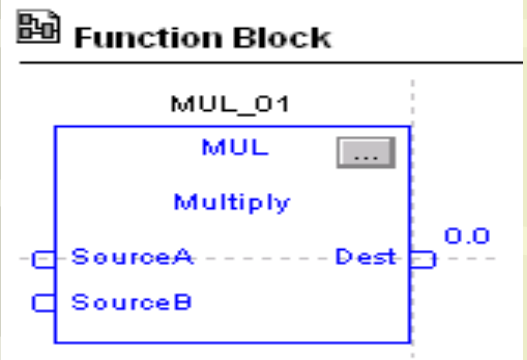
Instrução ADD - Adição

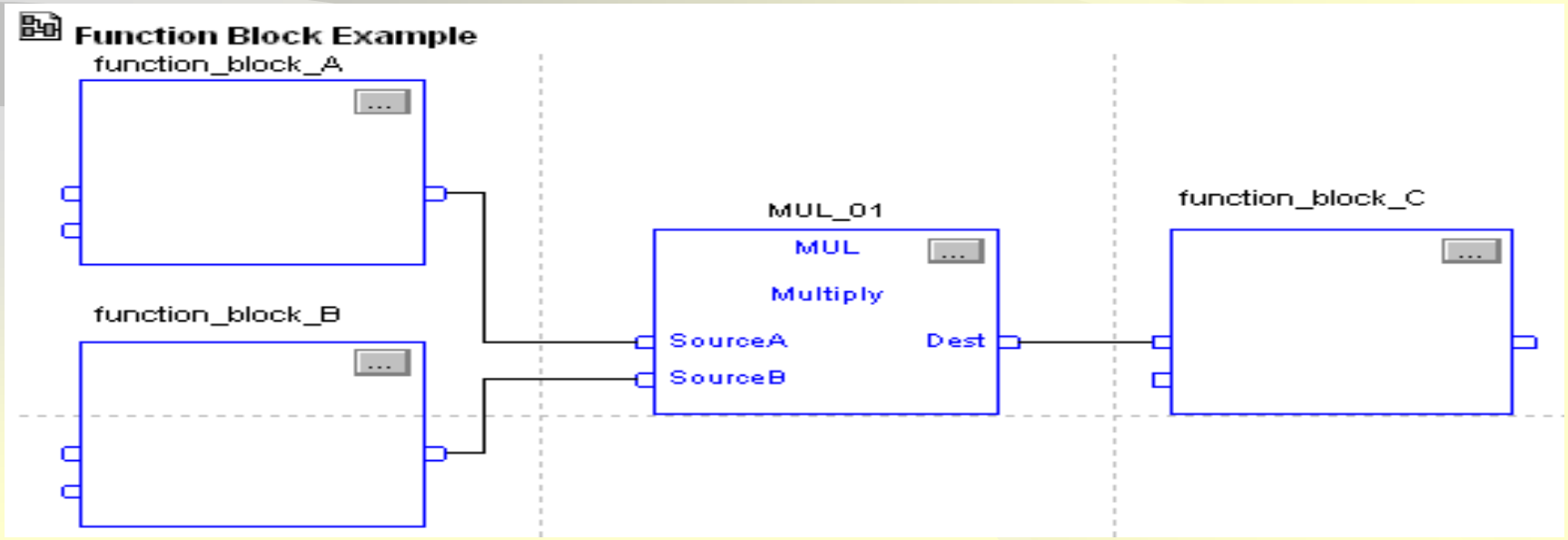




Aritméticas

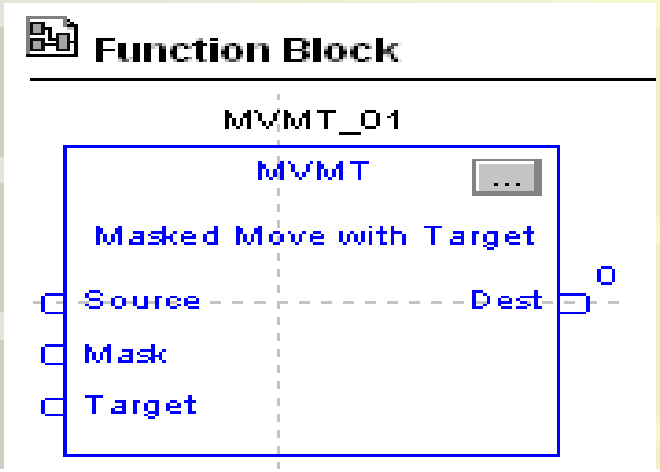
Instrução MUL - Multiplicação

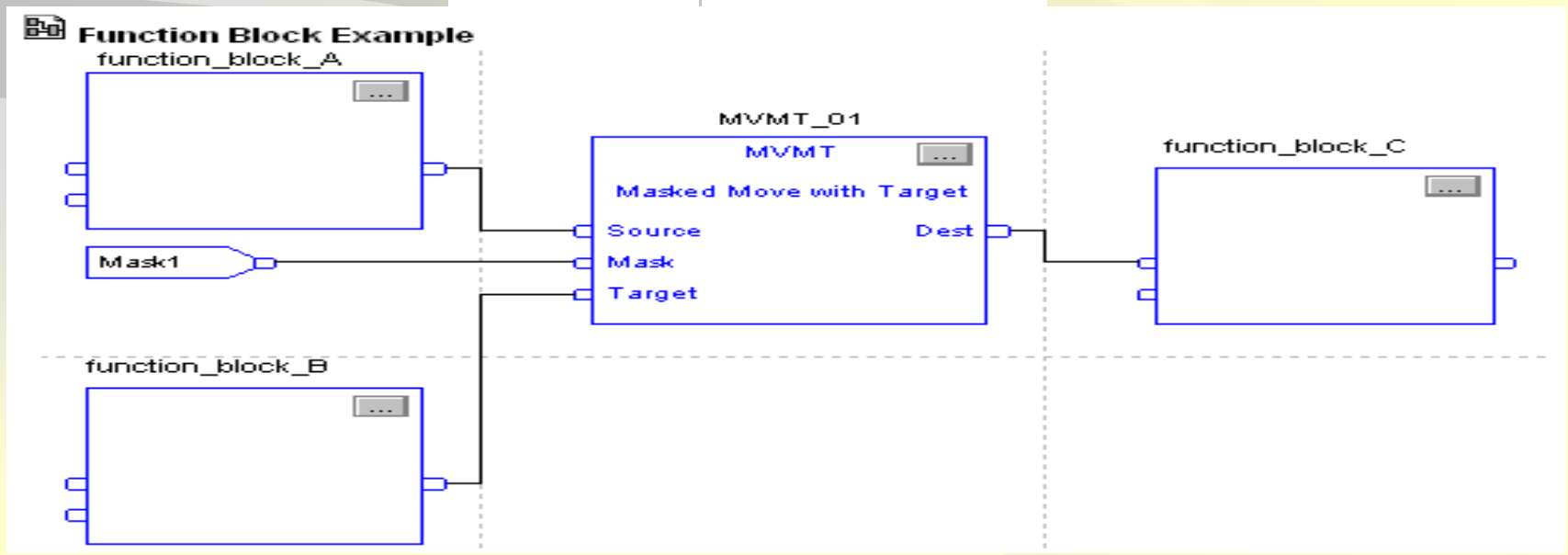




Manipulação de dados

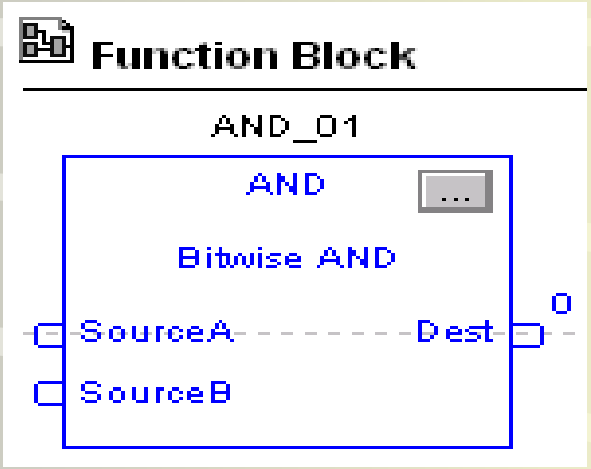
Instrução MVMT - Carregar dado e mover com máscara (MoVe with Mask and Target)

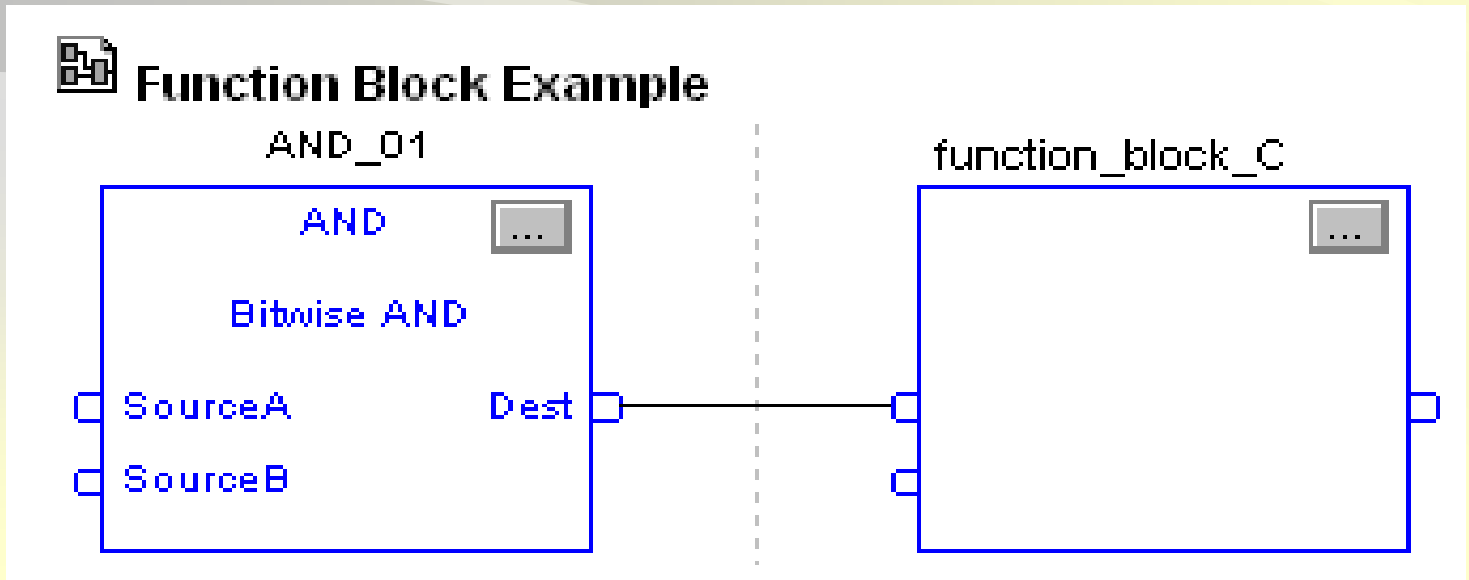




Manipulação de dados

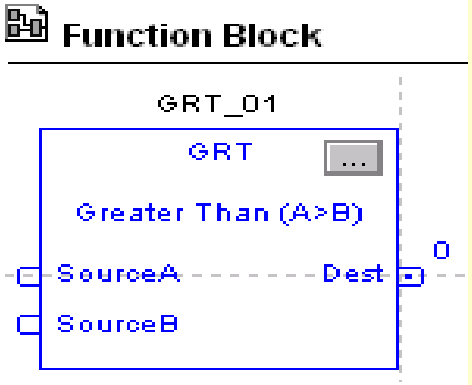
Instrução AND - E (AND)

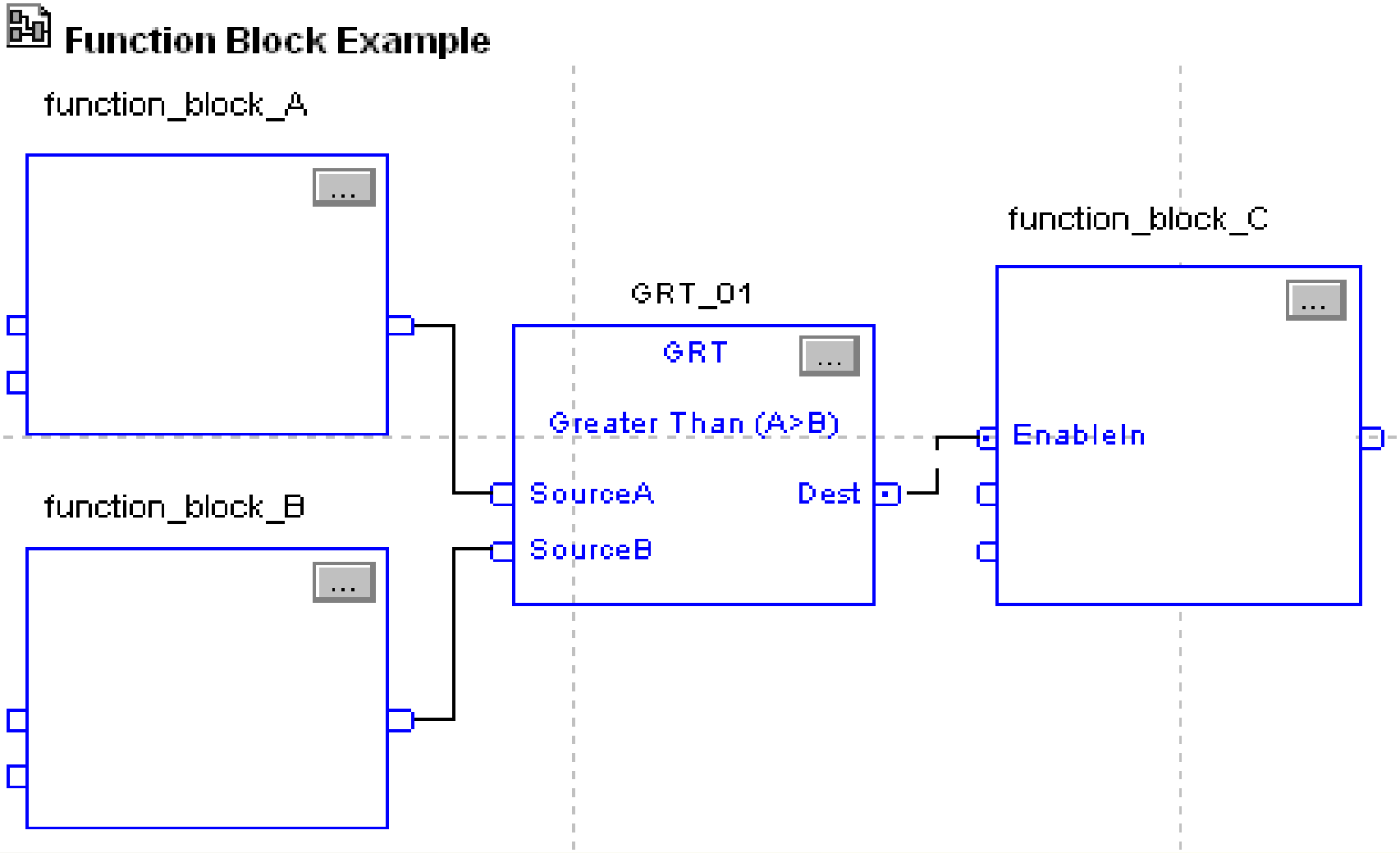




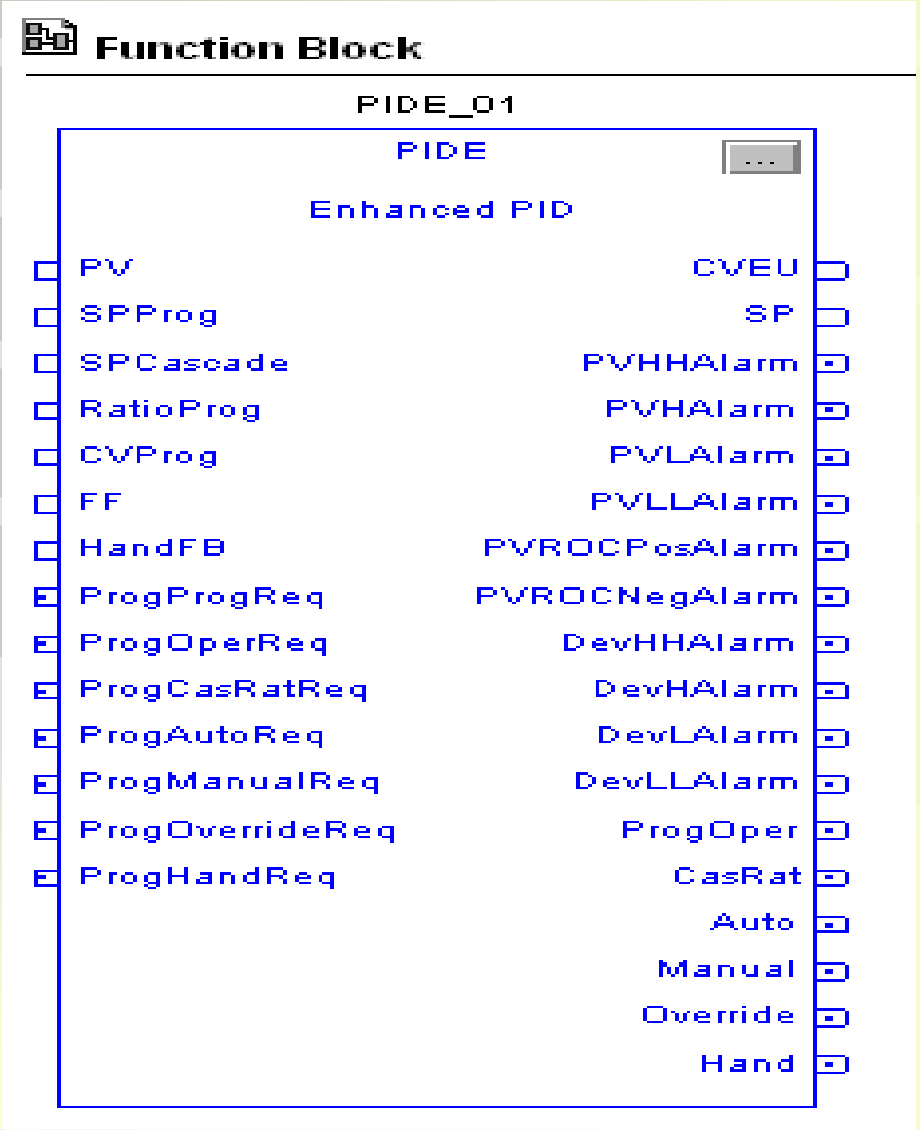
Comparação

Instrução GRT - Maior que (GReater Than)



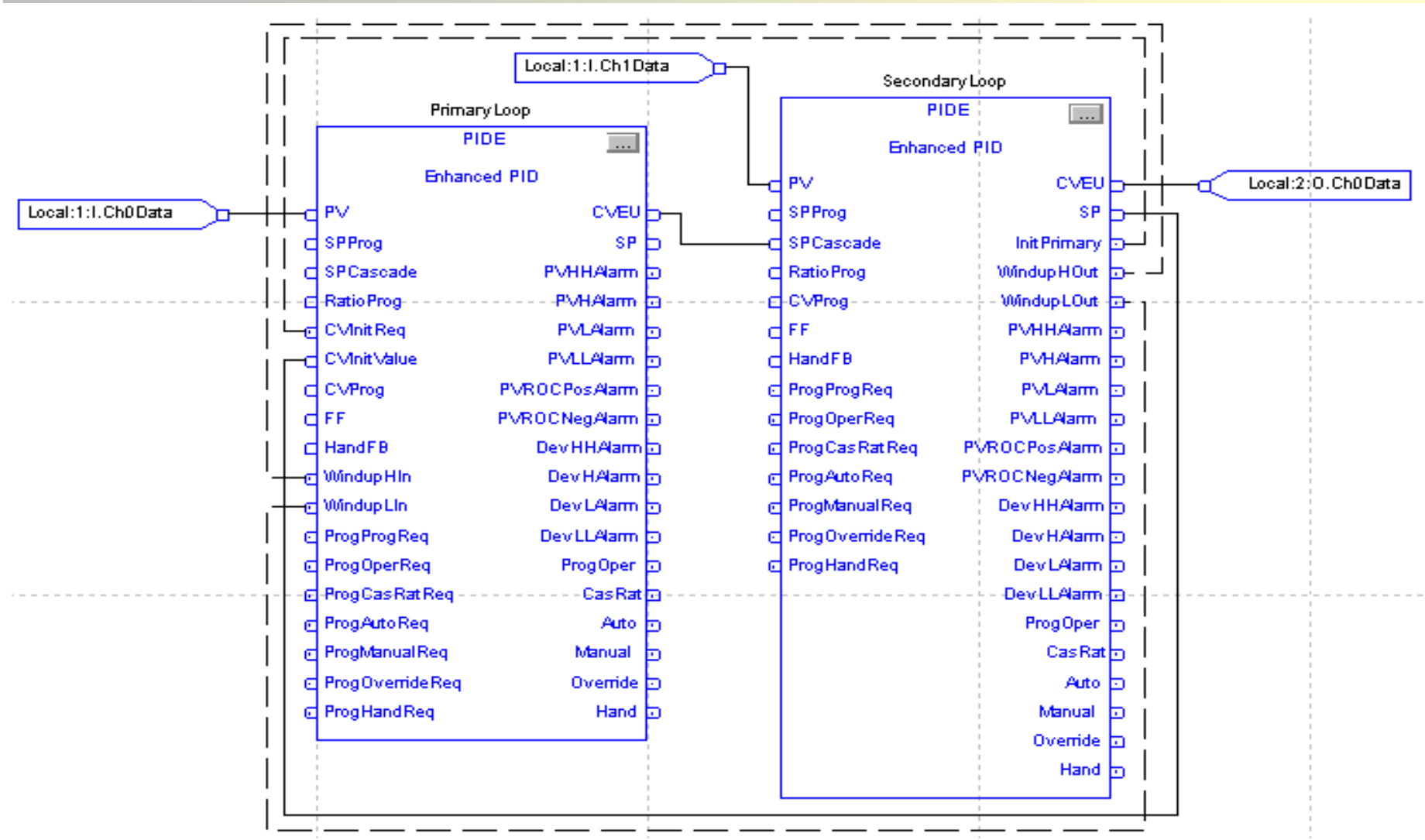


Instrução PID - Controle PID (Proporcional Integral Derivative)

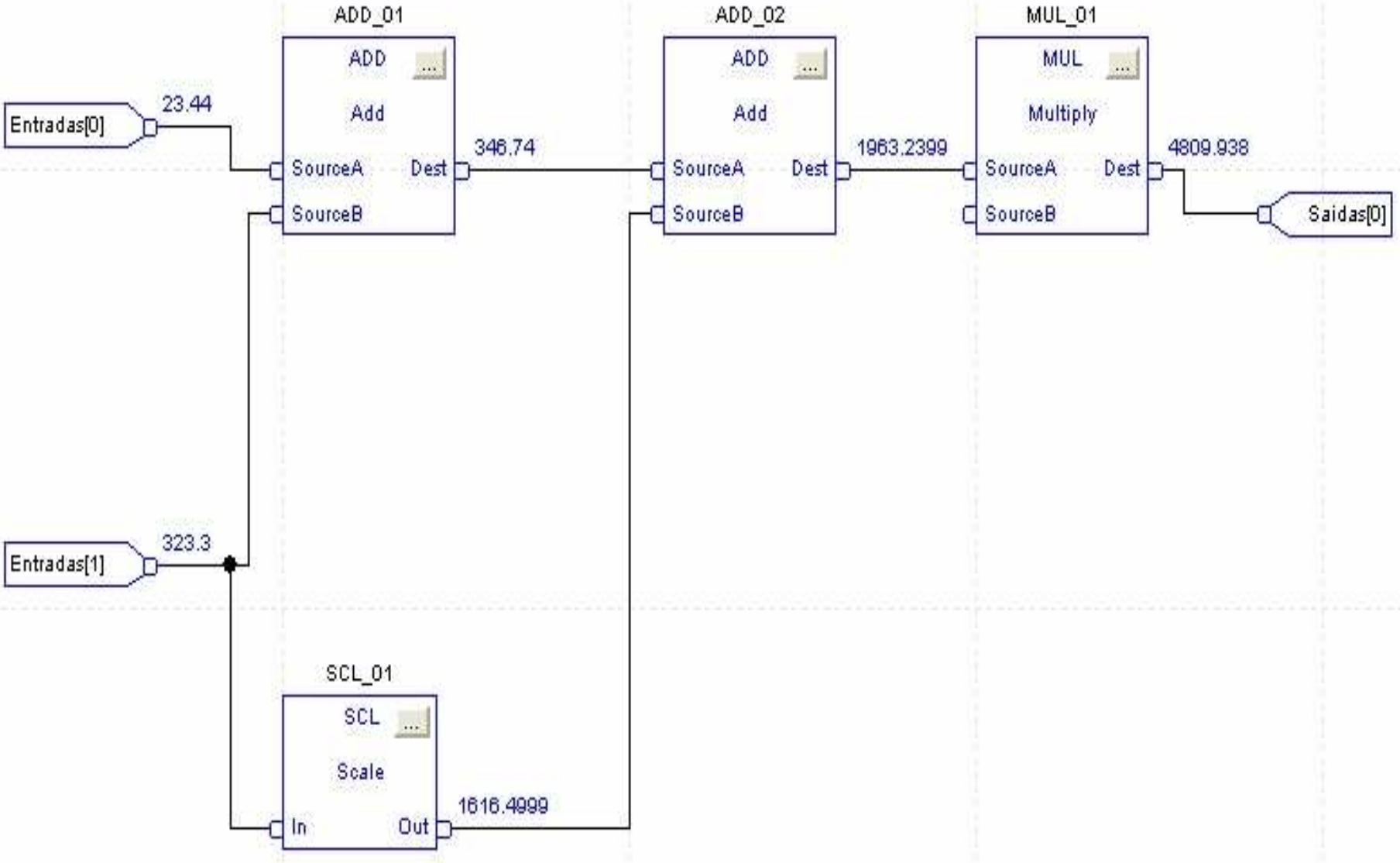


Avançadas

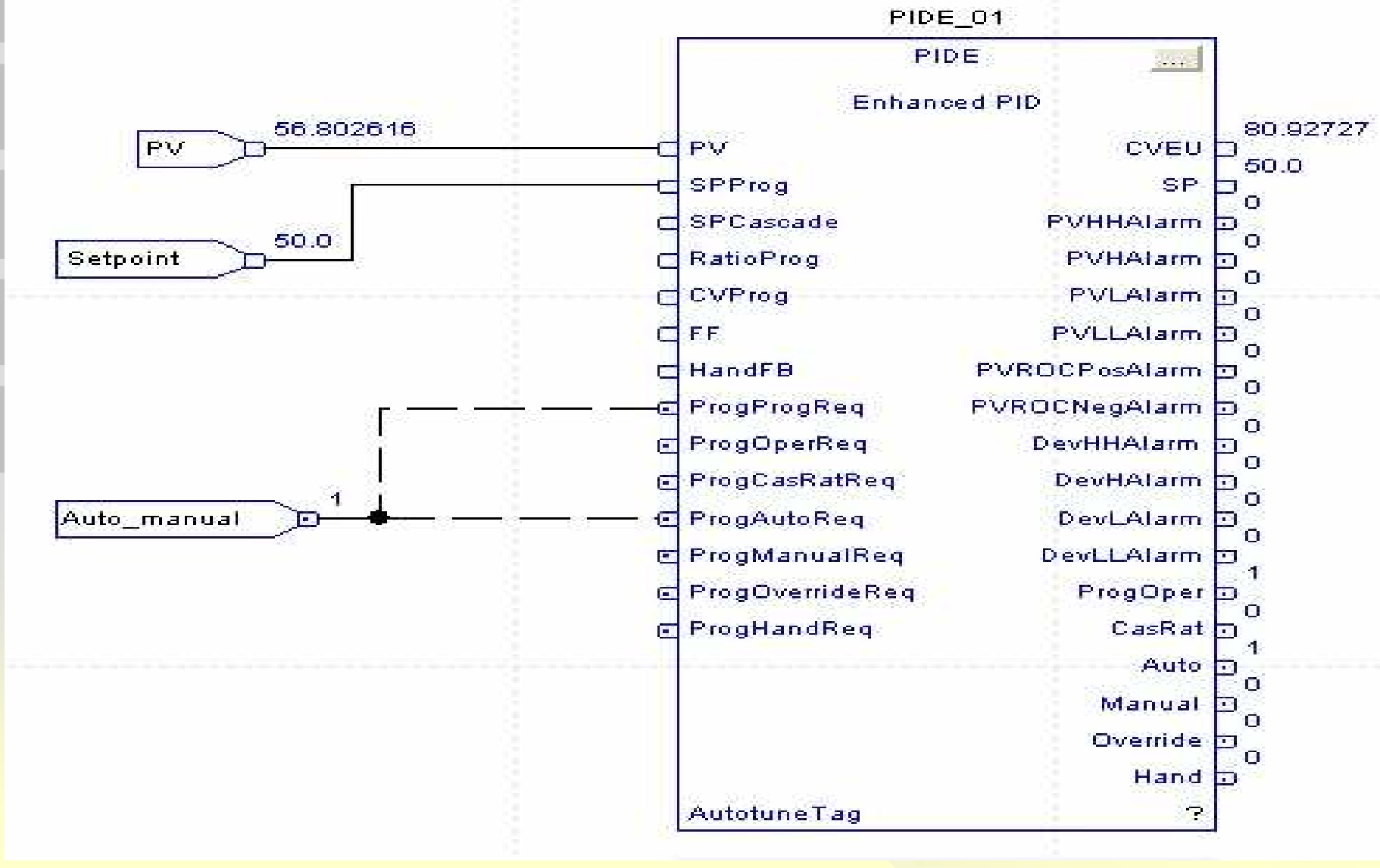
Instrução PID - Controle PID (Proporcional Integral Derivative)



Exemplo – Tratamento de Analógico (ADD, MUL, SCL)



Exemplos – Malha de Controle (PIDE)



Texto estrutura “Structured text” (ST):

Como a linguagem IL a linguagem ST é uma linguagem de alto nível porque não emprega operadores de baixo nível de máquina como uma linguagem do tipo “assembler”. Entretanto, oferece uma grande quantidade de declarações abstratas que podem descrever operações complexas de uma maneira simples..Um programa escrito em ST consistem de um número de declarações separadas por (;). A linguagem ST permite que os programas sejam estruturadoS

- Consiste em uma linguagem de programação de alto nível;

- Assemelha-se à linguagem Pascal na maneira como os códigos são escritos;

- A utilização desta linguagem permite a implementação de algoritmos com processamento de dados com loop´s em tabelas ou vetores de dados, o processamento de protocolos, o manuseio de strings ASCII e a realização de operações matemáticas complexas;

- Pode ser utilizado também para a criação de novos blocos funcionais.

Tipos de instruções

FIND	Localização de determinada string numa string
INSERT	Inserção de caracteres ASCII numa string
CONCAT	Concatenação de caracteres ASCII numa string
DELETE	Eliminação de caracteres ASCII numa string
PI	Controle tipo Proporcional Integral
INTG	Controle do tipo Integral
PID	Controle tipo Proporcional Integral Derivativo
COP	Cópia de valores
SIZE	Verificação de tamanho do vetor
TRUNC	Comando truncar resultado
SEL	Seleção de valores de entrada
STD	Cálculo de desvio padrão
AVE	Cálculo da média aritmética
IF...THEN	Execução de uma ação se determinada condição for verdadeira
CASE...OF	Seleção de execução em função do valor numérico
FOR...DO	Repetição num determinado intervalo de valor
WHILE...DO	Repetição enquanto uma determinada condição é verdadeira
REPEAT...UNTIL	Repetição até uma determinada condição se tornar verdadeira

Palavra-chave	Descrição	Exemplo	Explicação
:=	Declaração	D := 10	Declara o valor da direita para o identificador à esquerda
	Chama uma FB	Nome FB Parte1=10 Parte2=20	Chama uma outra POU ou tipo de FB incluindo seus parâmetros.
RETURN	Retorna	RETUNR	Deixa a corrente POU e para a POU chamada
IF	Seleção	IF d < e THEN f:=1; ELSE d=e THEN f:=2; ELSE f:=3; END IF	Seleciona alternativas por meio de expressões booleanas.
CASE	Seleção múltipla	CASE f OF 1: g:=11; 2: g:=12 ELSE ND CASE;	Seleção de blocos de declarações dependendo do valor da expressão.
FOR	Interação 1	FOR h:=1 TO 10 BY 2 DO F[h/2]:=h END_FOR	Vários loops com condicionantes de início e fim.

WHILE	Interação 2	WHILE m> 1 DO
		N := n/2
		END WHILE;
			Vário loops com condicionantes de início e fim no começo.

REPEAT	Interação 3	REPEAT I:=i* UNTIL i < 1000 END REPEAT	Vários loops com condicionante no fim.
EXIT	Fim do loop	EXIT	Terminações prematuras.
;	Declaração	; ;	

Declarações na linguagem ST (Karl and Michael Tiegelkamp)

Exemplo – Seleção (CASE ... OF)

```
(* Parametrização em Função do Programa Selecionado *)

CASE Selecao_Programa OF
(* Parametrização do PROGRAMA 1 *)
    1:      Sequencia_01 :=1;
            Sequencia_02 :=0;
            Sequencia_03 :=0;
(* Parametrização do PROGRAMA 2 *)
    2:      Sequencia_01 :=1;
            Sequencia_02 :=1;
            Sequencia_03 :=0;
(* Parametrização do PROGRAMA 3 *)
    3:      Sequencia_01 :=1;
            Sequencia_02 :=1;
            Sequencia_03 :=1;
ELSE
(* Sem seleção de PROGRAMA *)
            Sequencia_01 :=0;
            Sequencia_02 :=0;
            Sequencia_03 :=0;

END_CASE;
```

Exemplo – Loop de execução (WHILE ... DO)

```
(* Rotina para cálculo dos setpoints *)


Posicao := 0;

WHILE Posicao < 100 DO

Posicao := Posicao + 2;
Vetor_RESULTADOS[Posicao] := Medicao*Posicao;

END_WHILE;
```

REFERÊNCIAS:

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

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

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

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

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

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

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

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

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

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

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

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

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

Rockweel Automation Drives Engineering Handbook

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

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

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

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

Yamaguchi M. C. Módulo III – Controladores Programáveis – Apostila PECE EPUSP