

SENAI / CIMATEC

S i s t e m a F I E B

Área Tecnológica de Mecatrônica



CLP SIEMENS

SENAI / CIMATEC

S i s t e m a F I E B

Área Tecnológica Mecatrônica

SENAI CIMATEC

CLP SIEMENS

SALVADOR
2002

Copyright © 2002 por SENAI CIMATEC. Todos os direitos reservados.

Área Tecnológica de Mecatrônica

Elaboração: Milton Bastos de Souza

Revisão Técnica: Eduardo Lima II

Revisão Pedagógica: Maria Célia Calmon Santos

Normalização: Maria do Carmo Oliveira Ribeiro

Catalogação na fonte

SENAI CIMATEC – Centro Integrado de Manufatura
e Tecnologia. **CLP SIEMENS**. Salvador, 2002. 105.p.
il. (Rev.00)

I. Sistemas Digitais - CLP I. Título

CDD 621.3

SENAI CIMATEC

Av. Orlando Gomes, 1845 - Piatã

Salvador – Bahia – Brasil

CEP 41650-010

Tel.: (71) 462-9500

Fax. (71) 462-9599

<http://www.cimatec.fieb.org.br>

MENSAGEM DO SENAI CIMATEC

O SENAI CIMATEC visa desenvolver um programa avançado de suporte tecnológico para suprir as necessidades de formação de recursos humanos qualificados, prestação de serviços especializados e promoção de pesquisa aplicada nas tecnologias computacionais integradas da manufatura.

Com uma moderna estrutura laboratorial e um corpo técnico especializado, o CIMATEC desenvolve programas de intercâmbio tecnológico com instituições de ensino e pesquisa, locais e internacionais.

Tudo isso sem desviar a atenção das necessidades da comunidade, atendendo suas expectativas de formação profissional, suporte tecnológico e desenvolvimento, contribuindo para uma constante atualização da indústria baiana de manufatura e para a alavancagem do potencial das empresas existentes ou emergentes no estado.

SENAI CIMATEC

APRESENTAÇÃO

O curso de Controladores Lógicos Programáveis, objetiva fornecer os subsídios necessários para que o treinando, esteja a par do atual estágio tecnológico da instrumentação, aplicada ao controle de processos.

Este material enfoca aspectos gerais relacionados ao CLP, com a intenção de ser o mais genérico possível.

Não pretendemos com este curso esgotar o tema, mas indicar o caminho, para os que irão atuar na manutenção de equipamentos digitais e projetos de pequenos sistemas de automação utilizando CLPs aplicados no controle de variáveis de processos industriais.

Esperamos que os treinandos tirem o maior proveito deste material, pois, ele é uma síntese do conhecimento de vários especialistas na área de Sistemas Digitais.

SUMÁRIO

1. Histórico dos CLPs.....	8
2. Estrutura Básica de um CLP.....	11
2.2 Entradas	11
2.2 Saída.....	14
2.3 Unidade Central de Processamento (Central Processing Unit – CPU).....	16
2.4 O que caracteriza o tamanho do CLP?	17
2.5 Família de Hardware para PLCs Siemens	17
2.5.1 Características das CPUs	17
3. Introdução a Liguagem Siemens.....	30
3.1 Ladder	30
3.2 STL	36
4. Instruções avançadas da Siemens	44
4.2 Instruções Matemáticas	46
4.2.1 Exemplos de Casos para PLCs Allen-Bradley	46
4.2.2 Estudo de Casos para PLCs Siemens	49
4.3 Instruções de Comparação	55
4.3.1 Estudo de Casos para PLCs da Allen-Bradley.....	55
4.3.2 Estudos de Casos para o CLP Siemens	59
4.4 Instruções de Controle de Programa.....	61
4.5 Chamada de Subrotina	66
4.5.1 Para PLC Allen-Bradley.....	66
4.5.2 Chamada de Subrotina para PLC SIEMENS.....	67
4.6 SCL – Escala	72
4.6.1 Escala para CLP Allen-bradley.....	72
4.6.2 Escala em CLP SIEMENS	74
4.6.3 UNSCALE para CLP SIEMENS	74
5 INSTRUÇÃO PID.....	76
5.1 Conceito de PID.....	76
5.2 A Equação PID	77
5.3 Parâmetros para o PID Allen-Bradley	78
5.4 Tela de Instalação PID	79
5.5 Bloco de Controle PID	81
5.6 Indicadores de Status	82
5.7 Bloco PID para PLCs SIEMENS.....	83
6 Movimentação de Dados.....	87
6.1 Copiar Arquivo.....	87
6.1.1 Copiar arquivo CLP Allen-Bradley	87
6.1.2 Copia arquivo Pelo CLP SIEMENS(FC81).....	88

6.2 Preencher Arquivo.....	88
6.2.1 Para CLP Allen-Bradley(FLL).....	88
6.2.2 Preenchimento para CLP SIEMENS(SFC21).....	89
6.3 Deslocar Bit à Direita e Deslocar Bit à Esquerda.....	90
6.3.1 Deslocar Bit para o CLP Allen-bradley	90
6.3.2 Desloca Bits para o CLP SIEMENS.....	91
ANEXO 1.....	95
ANEXO 2.....	104
BIBLIOGRAFIA.....	105

SENNA CLIMATEC

1. Histórico dos CLPs

Segundo a NEMA (National Electrical Manufacturers Association), o Controlador Lógico programável (CLP) é definido como aparelho eletrônico digital que utiliza uma memória programável para o armazenamento interno de instruções específicas, tais como lógica, sequenciamento, temporização, contagem e aritmética, para controlar, através de módulos de entradas e saídas, vários tipos de máquinas e processos.

O desenvolvimento dos CLPs começou em 1968 em resposta a uma requisição da Divisão Hidráulica da General Motors. Naquela época, a General Motors passava dias ou semanas alterando sistemas de controles baseados em relés, sempre que mudava um modelo de carro ou introduzia modificações em uma linha de montagem. Para reduzir o alto custo de instalação decorrente destas alterações, a especificação de controle da GM necessitava de um sistema de estado sólido, com a flexibilidade de um computador, mas que pudesse ser programado e mantido por engenheiros e técnicos na fábrica. Também era preciso que suportasse o ar poluído, a vibração, o ruído elétrico e os extremos de umidade e temperatura encontrados normalmente num ambiente industrial. Abaixo alguns modelos de CLPs.



Os primeiros CLPs foram instalados em 1969, fazendo sucesso quase imediato. Funcionando como substitutos de relés, até mesmo estes primeiros CLPs eram mais confiáveis do que os sistemas baseados em relés, principalmente devido à robustez de seus componentes de estado sólido quando comparados às peças móveis dos relés eletromecânicos. Os CLPs permitiram reduzir os custos de materiais, mão-de-obra, instalação e localização de falhas ao reduzir a necessidade de fiação e os erros associados. Os CLPs ocupavam menos espaço do que os contadores, temporizadores e outros componentes de controle anteriormente utilizados.

E a possibilidade de serem reprogramados permitiu uma maior flexibilidade para trocar os esquemas de controle.

Talvez a razão principal da aceitação dos CLPs pela indústria foi que a linguagem inicial de programação era baseada nos diagramas de contato (ladder) e símbolos elétricos usados normalmente pelos eletricistas. A maior parte do pessoal de fábrica já estava treinada em lógica ladder, adaptando-a rapidamente nos CLPs.

Por que usar um CLP?

“Deveríamos estar usando um controlador lógico programável?” Nos anos 70 e inicio dos 80, muitos engenheiros, gerentes de fábrica e projetistas de sistema de controle dedicaram grande parte de seu tempo a debater esta questão, tentando avaliar a relação custo-benefício.

Atualmente, aceita-se como regra geral que os CLPs se tornaram economicamente viáveis nos sistemas de controle que exigem mais de três relés. Considerando-se o baixo custo dos micro-CLPs e o fato dos fabricantes colocarem grande ênfase na qualidade e produtividade, a questão do custo deixa praticamente de existir. Além das reduções nos custos, os CLPs oferecem outros benefícios de valor agregado:

- ⇒ *Confiabilidade.* Depois de escrito e depurado, um programa pode ser transferido e armazenado facilmente em outros CLPs. Isto reduz o tempo de programação, minimiza a depuração e aumenta a confiabilidade. Como toda a lógica existe na memória do CLP, não existe qualquer possibilidade de cometer um erro lógico por conta de um erro de fiação. A única fiação necessária é para o fornecimento de energia para as entradas e saídas.
- ⇒ *Flexibilidade.* As modificações no programa podem ser feitas com pouca digitação. Os OEMs (fabricantes do equipamento original) podem realizar facilmente as atualizações no sistema, bastando enviar um novo programa em vez de um técnico. Os usuários finais podem modificar o programa em campo ou, por outro lado, os OEMs podem evitar que os usuários finais alterem o programa (o que é uma importante característica de segurança).
- ⇒ *Funções Avançadas.* Os CLPs podem realizar uma grande variedade de tarefas de controle, desde ações simples e repetitivas até a manipulação de dados complexos. Com a adoção dos CLPs, abrem-se muitas alternativas para os projetistas e simplifica-se o trabalho do pessoal de manutenção.
- ⇒ *Comunicações.* A comunicação com interfaces de operação, outros CLPs ou computadores facilita a coleta de dados e o intercâmbio de informações.
- ⇒ *Velocidade.* Como certas máquinas automatizadas processam milhares de itens por minuto e como os objetos são expostos aos sensores durante apenas uma fração de segundo, muitas aplicações de automação necessitam da capacidade de resposta rápida dos CLPs.

⇒ *Diagnóstico.* A capacidade de localização de falhas dos dispositivos de programação e o recurso de diagnóstico incorporado no CLP permitem que os usuários localizem e corrijam rapidamente os problemas de software e hardware.

Outras Características

- ⇒ Hardware e/ou dispositivo de controle de fácil e rápida programação ou reprogramação, com a mínima interrupção na produção.
- ⇒ Capacidade de operação em ambiente industrial sem o apoio de equipamentos ou hardware específicos.
- ⇒ Sinalizadores de estado e módulos tipo plug-in de fácil manutenção e substituição.
- ⇒ Hardware ocupando espaço reduzido e apresentando baixo consumo de energia.
- ⇒ Possibilidade de monitoração do estado e operação do processo ou sistema, através da comunicação com computadores.
- ⇒ Compatibilidade com diferentes tipos de sinais de entrada e saída.
- ⇒ Capacidade de alimentar, de forma contínua ou chaveada, cargas que consomem correntes de até 2 A.
- ⇒ Hardware de controle que permite a expansão dos diversos tipos de módulos, de acordo com a necessidade.
- ⇒ Custo de compra e instalação competitivo em relação aos sistemas de controle convencionais.
- ⇒ Possibilidade de expansão da capacidade de memória.
- ⇒ Conexão com outros CLPs através de redes de comunicação

Aplicações Tradicionais

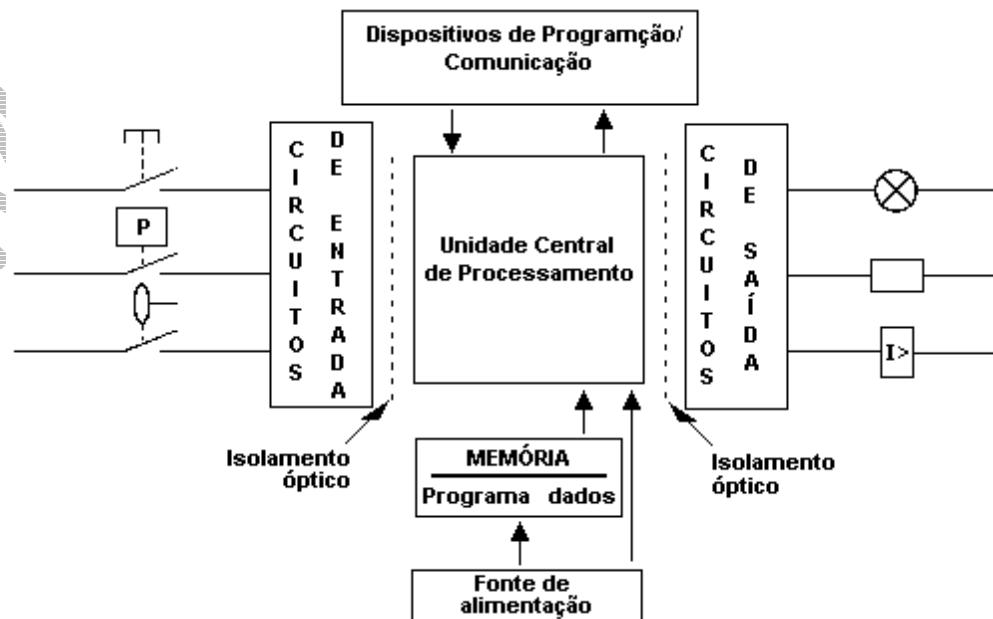
Seja qual for a aplicação, o uso do CLP permite aumentar a competitividade. Os processos que usam CLPs incluem: empacotamento, engarrafamento e enlatamento, transporte e manuseio de materiais, usinagem, geração de energia, sistemas de controle predial e de ar condicionado, sistemas de segurança, montagem automatizada, linha de pintura e tratamento de água. Os CLPs são utilizados nas mais diversas indústrias, incluindo alimentos e bebida, automotiva, química, plásticos, papel e celulose, farmacêutica e siderurgia/metalurgia. Basicamente qualquer aplicação que exija um controle elétrico pode usar um CLP.

2. Estrutura Básica de um CLP

A Estrutura básica de um controlador programável adveio do hardware básico de um computador. Podemos afirmar que um CLP é um computador para aplicações específicas.

Para entender como funciona um CLP, é necessária uma análise rápida de seus componentes. Todos os CLPs, dos micros aos grandes CLPs, usam os mesmos componentes básicos e estão estruturados de forma similar, como mostrado na figura abaixo. Os sistemas CLP consistem de :

1. Entradas
2. Saídas
3. Unidade Central de Processamento (Central Processing Unit – CPU)
4. Memória para o programa e armazenamento de dados
5. Fornecimento de alimentação
6. Dispositivo de programação



2.2 Entradas

Os terminais de entrada conectados no CLP formam a interface pela qual os dispositivos de campo são conectados ao CLP.

Os sinais recebidos por um módulo de entrada podem vir de dois tipos de sensores:

⇒ **Discretos:**

Chave limite; botoeira; chave de digitadora (thumbwheel); chave de pressão; fotocélula; contato de relé; chave seletora; teclado.

⇒ **Analógicos:**

Transdutor de pressão; transdutor de temperatura; célula de carga (strain gage); sensores de vazão; transdutores de vibração; transdutores de corrente; transdutores de vácuo; transdutores de força.

Os sinais elétricos enviados pelos dispositivos de campo ao CLP são normalmente de 120Vca ou de 24Vcc. Os circuitos de entrada no CLP recebem esta tensão vinda do campo e a “condicionam” de forma que possa ser utilizada pelo CLP. Tal condicionamento é necessário já que os componentes internos de um CLP operam a 5Vcc e devem, portanto, estar protegidos de flutuações de tensão. Para que os componentes internos fiquem eletricamente isolados dos terminais de entrada, os CLPs empregam um isolador óptico que usa a luz para acoplar os sinais de um dispositivo elétrico a outro.

A estrutura interna de um módulo de entrada pode ser subdividida em seis blocos principais, como mostrado na figura abaixo:

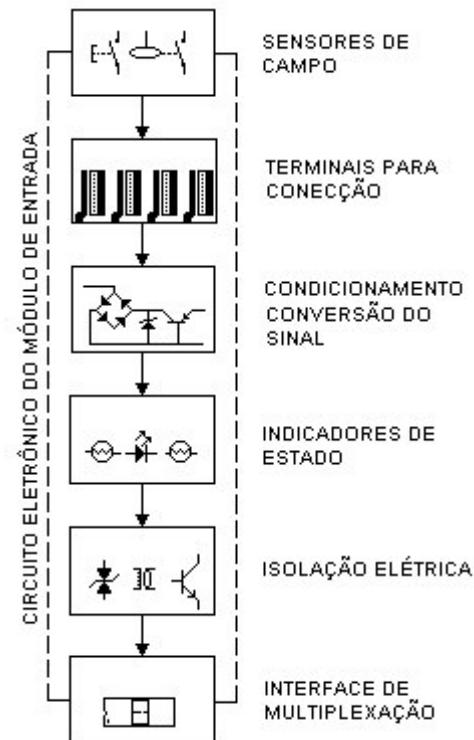


Tabela onde podemos ver a função de cada bloco:

Parte	Função
Sensores de campo	Informar ao controlador programável as condições do processo
Terminais para conexão dos sensores de campo	Permitir a interligação física entre os sensores de campo e o controlador programável.
Condicionamento e conversão do sinal de entrada	Converter os sinais de campo em níveis baixos de tensão, compatíveis com o processador utilizado.
Indicadores de estado das entradas	Proporcionar indicação visual do estado funcional das entradas contidas num módulo de entrada.
Isolação elétrica	Proporcionar isolação elétrica entre os sinais vindos do campo e os sinais do processador.
Interface/multiplexação	Informar ao processador o estado de cada variável de entrada.

Dependendo da natureza do sinal de entrada, podemos dispor dos seguintes tipos de módulos de entrada:

TIPO	CARACTERÍSTICAS
DIGITAL (AC)	12 Vac; 24 A 48 Vac; 110/127 Vac; 220/240 Vac
DIGITAL (DC)	120 Vdc com isolação 12 Vdc; 12 a 24 Vdc com resposta rápida; 24 a 48 Vdc; 12 a 24 Vdc (lógica negativa) source; 12 a 24 Vdc (lógica positiva) sinking; 48 Vdc source; 48 Vdc sinking
ANALÓGICO	1 a 5 Vdc; 0 a 10Vdc; -10 a +10Vdc; 4 a 20mA.
ESPECIAL	TTL com suprimento; TTL com dreno; 5 a 30 Vdc selecionável; 5Vdc contador/ decodificador; 12 a 24Vdc codificador/ contador; termopar; código ASCII; código Gray; pulsos de alta velocidade.

2.2 Saída

Os módulos de saída também são considerados como elementos de interface, pois permitem que o processador se comunique com o meio externo. A estrutura interna de um módulo de saída pode ser subdividida em sete blocos principais, relacionados a seguir:

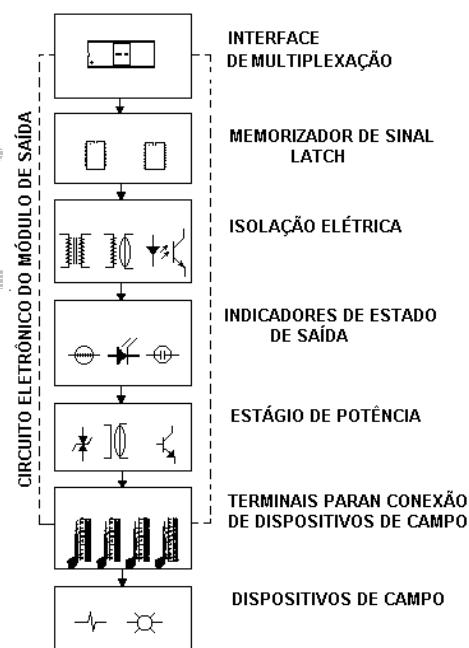


Tabela onde podemos ver a função de cada bloco:

Parte	Função
Interface/Demultiplexação	Recebe os sinais vindos do processador direcionando-os para as respectivas saídas.
Memorizador de sinal	Armazena os sinais que já foram multiplexados pelo bloco anterior.
Isolação elétrica	Proporciona isolamento elétrico entre os sinais vindos do processador e os dispositivos de campo.
Indicadores de estado de saídas	Proporciona indicação visual do estado funcional das saídas contidas num módulo de saída
Estágio de Potência	Transforma os sinais lógicos de baixa potência vindos do processador em sinais de potência, capazes de operar os diversos tipos de dispositivos de campo
Terminais para conexão dos dispositivos de campo	Permite a conexão física entre CLP e os dispositivos de campo.
Dispositivos de campo	Consiste em dispositivos eletromecânicos que atuam no processo/equipamento, em função dos sinais de controle enviados pelo CP.

Dependendo da natureza dos dispositivos de campo e do tipo de sinal de controle necessário para comandá-los, podemos dispor dos seguintes tipos de módulos de saída:

TIPO	CARACTERÍSTICAS
Digital (AC)	12Vac; 24 a 48Vac; 120Vac; 220/240Vac; 120Vac com isolação.
Digital (DC)	12 a 60Vdc; 12 a 24Vdc com resposta rápida; 24 a 48Vdc; 12 a 24Vdc com suprimento; 12 a 24Vdc com dreno; 48Vdc com suprimento; 48Vdc com dreno.
Analógico	1 a 5Vdc; 0 a 10Vdc; -10 a +10Vdc; 4 a 20mA.
Especial	TTL com suprimento; TTL com dreno; 5 a 30Vdc selecionável; contato NA; contato NF; saída em ASCII; servo-motor; motor de passo.

Os circuitos de saída funcionam de maneira similar aos circuitos de entrada: os sinais emitidos pela CPU passam por uma barreira de isolamento antes de energizar os circuitos de saída.

Os CLPs utilizam vários circuitos de saída para energizar seus terminais de saída: relés, transistores e triacs.

- ⇒ *Relés.* Os Relés podem ser usados com alimentação alternada ou contínua. Os relés eletromagnéticos de CLPs tradicionais aceitam correntes de até alguns ampéres. Os relés suportam de forma melhor os picos de tensão porque contém uma camada de ar entre seus contatos que elimina a possibilidade de ocorrência de fuga. No entanto, são comparativamente lentos e sujeitos a desgaste com o tempo.
- ⇒ *Transistores.* Os transistores chaveiam corrente contínua, são silenciosos e não contém peças móveis sujeitas a desgaste. Os transistores são rápidos e podem reduzir o tempo de resposta, mas suportam cargas de, no máximo, 0,5A. Certos tipos especiais de transistores, os FETs (Transistores de Efeito de Campo) podem aceitar cargas maiores, normalmente de 1A.
- ⇒ *Triacs.* Os triacs chaveiam exclusivamente corrente alternada. Como os transistores, as saídas triacs são silenciosas, não tem peças móveis sujeitas a desgaste, são rápidas e transportam cargas de até 5A.

Obs. As saídas de estado sólido (triacs e transistor) podem ser danificadas e destruídas em caso de sobre tensão ou sobrecarga.

Os módulos de saída podem acionar os seguintes tipos de dispositivos de saída:

Discretos:

Controladores de motores, indicadores de painel, contator, válvula solenóide, display, bobina de relé, sistemas de alarme e segurança, sirena.

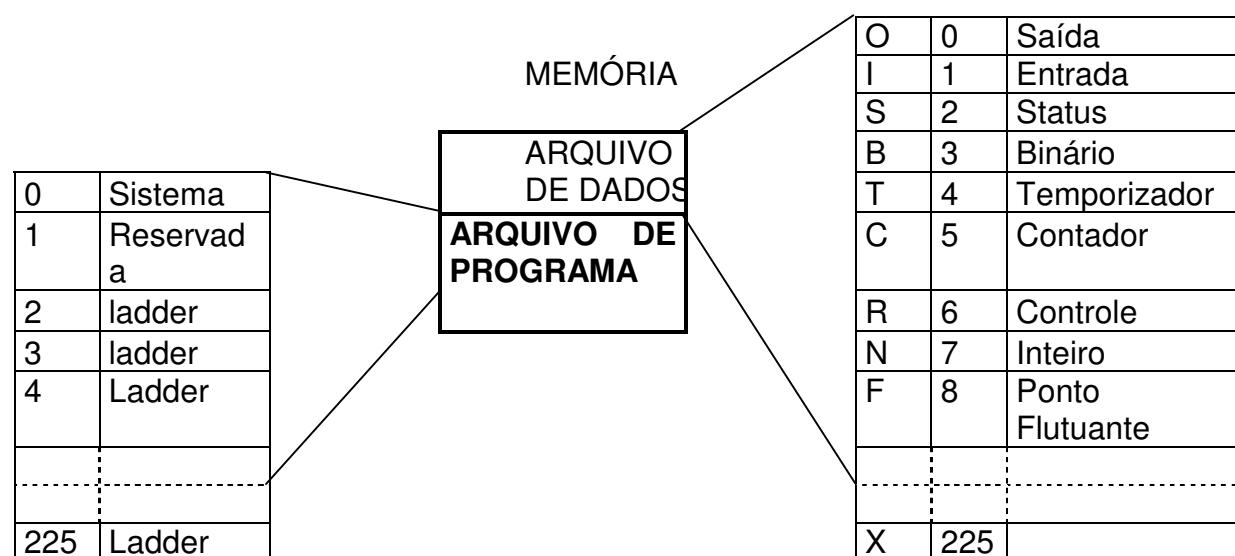
Analógicos:

Acionadores AC, válvula de controle, acionadores DC.

2.3 Unidade Central de Processamento (Central Processing Unit – CPU)

A CPU, formada por um microprocessador e um sistema de memória, é o principal componente do CLP. A CPU lê as entradas, executa a lógica segundo as instruções do programa de aplicação, realiza cálculos e controla as saídas, respectivamente.

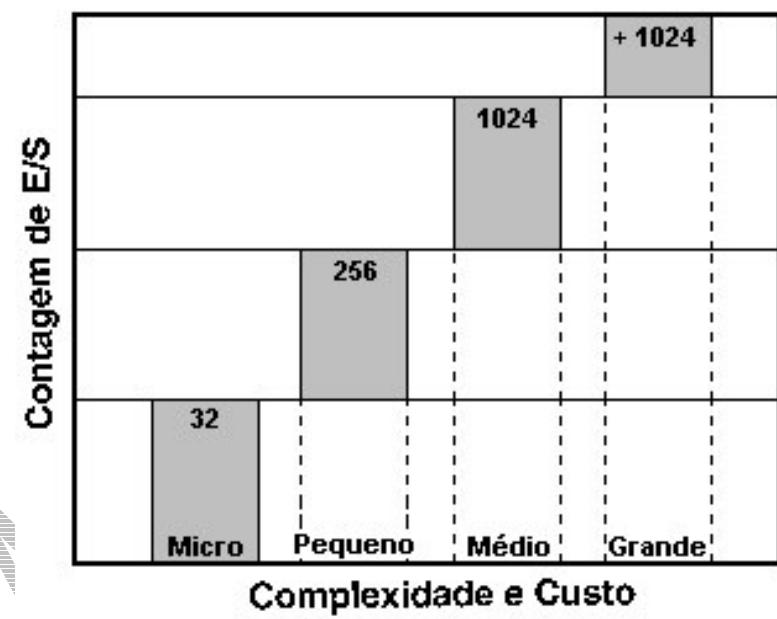
Os usuários de CLPs trabalham com duas áreas da CPU: *Arquivo de Programas* e *Arquivo de Dados*. Os Arquivos de Programa contêm o programa de aplicação do usuário, os arquivos de sub-rotina e as rotinas de falha. Os Arquivos de Dados armazenam dados associados com o programa, tais como status (condição) de entrada e saída, valores predefinidos e acumulados de contadores/temporizadores e outras constantes e variáveis. Juntas, estas duas áreas são chamadas de memória de aplicação ou memória do usuário. Veja a figura abaixo:



Ainda dentro da CPU encontra-se um programa executável ou *Memória do Sistema* que direciona e realiza as atividades de “operação”, tais como a execução do programa do usuário e a coordenação de varreduras das entradas e atualizações das saídas. A memória do sistema, programada pelo fabricante, não pode ser acessada pelo usuário.

2.4 O que caracteriza o tamanho do CLP?

Vários critérios são utilizados para classificar um CLP como micro, pequeno, médio ou grande, entre eles: funcionalidade, número de entradas e saídas, custo e dimensões físicas.



Os CLPs podem ser de *Estrutura Fixa* ou *Estrutura Modular*.

- ⇒ *Estrutura Fixa*. São unidades que já incluem o processador, a fonte de alimentação e as E/S reunidas em um só bloco.
- ⇒ *Estrutura Modular*. É aquele que tem componentes separados, porém interligados e podem ser expandidos com o acréscimo de mais módulos de E/S no chassi.

2.5 Família de Hardware para PLCs Siemens

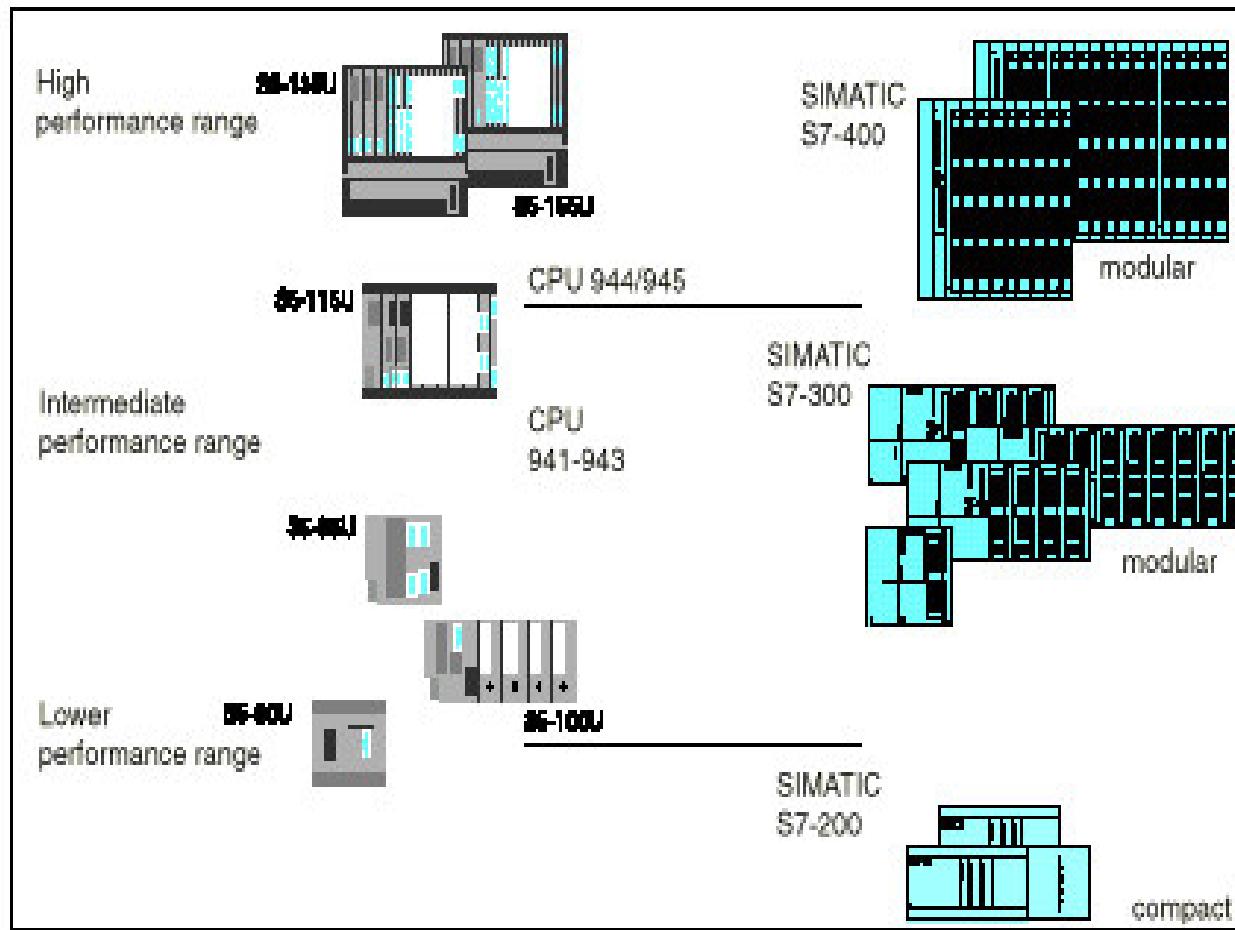
2.5.1 Características das CPUs

O SIMATIC S7 consiste dos seguintes tipos de PLCs, classificados de acordo a suas performance:

SIMATIC S7-200 é um PLC compacto MicroPLC, usados para aplicações que tenham baixíssimas performances.

SIMATIC S7-300 é um minecontrolador modular designado para aplicações de média performance.

SIMATIC S7-400 é designado para aplicações de media para alta performance.



SENAC/UFSCAR

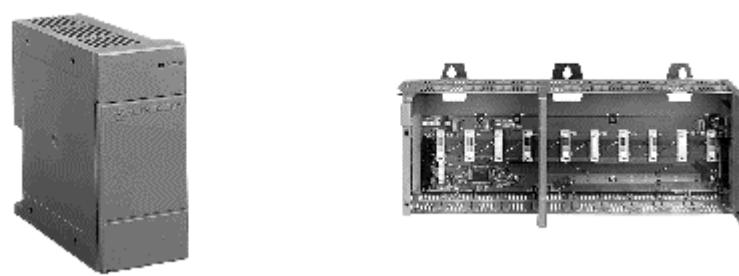
Características das CPUs Siemens

CPU S7-300	Descrição	MLFB
CPU Standard	CPU 313, 12KB, máx. 256 I/O; 0.6µs	6ES7313-1AD03-0AB0
	CPU 314, 24KB, máx. 1024 I/O; 0.3µs	6ES7314-1AE04-0AB0
	CPU 315, 48KB, máx. 1024 I/O; 0.3µs	6ES7315-1AF03-0AB0
	CPU 315-2 DP, 64KB, máx. 8192 I/O, incl DP; 0.3µs	6ES7315-2AF03-0AB0
	CPU 316-2DP, 128KB, máx. 16384 I/O, incl DP M/S; 0.3µs	6ES7316-2AG00-0AB0
	CPU 318-2DP, 512KB, máx. 65536 I/O, MPI 12 Mbits/s; incl DP M/S; 0.1µs	6ES7318-2AJ00-0AB0

CPU	CPU 312, 16KB, máx.256 I/O; 0.2µs	Cartão MMC requerido	6ES7312-1AD10-0AB0
Standard	CPU 314, 48KB, máx.1024 I/O; 0.1µs	Cartão MMC requerido	6ES7314-1AF10-0AB0
Nova Geração	CPU 315-2 DP, 128KB, máx. 1024 I/O,incl DP; 0.1µs	Cartão MMC requerido	6ES7315-2AG10-0AB0
CPU	CPU312C, 16KB, 10ED, 6SD, 2 cnt (10kHz), 0.2 µs	Cartão MMC requerido	6ES7312-5BD00-0AB0
Compacta	CPU313C, 32KB, 24ED, 16SD, 4+1EA,2SA, 3cnt(30kHz), 0.1µs	Cartão MMC requerido	6ES7313-5BE00-0AB0
	CPU313C-2 PtP, 32KB, 16ED,16SD,3cnt(30k Hz), PtP, 0.1µs	Cartão MMC requerido	6ES7313-6BE00-0AB0
	CPU313C-2 DP, 32KB, 16ED,16SD,3cnt(30k Hz), DP, 0.1µs	Cartão MMC requerido	6ES7313-6CE00-0AB0
	CPU314C-2 PtP, 48KB, 24ED, 16SD, 4+1 EA, 2SA, 4 cnt. (60kHz), PtP, 0.1µs	Cartão MMC requerido	6ES7314-6BF00-0AB0
	CPU314C-2DP, 48KB, 24ED, 16SD, 4+1EA, 2SA, 4 cnt. (60kHz), DP, 0.1µs	Cartão MMC requerido	6ES7314-6CF00-0AB0
CPU Fail-Safe	CPU 315F-2DP; 128KB.	Cartão MMC requerido	6ES7315-6FF00-0AB0
MMC	Micro cartão de memória 64 kbytes	Ver Nota *****	6ES7953-8LF00-0AA0
	Micro cartão de memória 128 kbytes	Ver Nota *****	6ES7953-8LG00-0AA0
	Micro cartão de memória 512 kbytes	Ver Nota *****	6ES7953-8LJ00-0AA0
	Micro cartão de memória 2 Mbytes	Ver Nota *****	6ES7953-8LL00-0AA0
	Micro cartão de memória 4 Mbytes	Ver Nota *****	6ES7953-8LM00-0AA0

Feature	CPU 312 IFM	CPU 313	CPU 314	CPU 314 IFM	CPU 315	CPU 315-2 DP
Work memory (integrated)	6 Kbytes	12 Kbytes	24 Kbytes	24 Kbytes	48 Kbytes	
Load memory						
• integrated	20 Kbytes RAM; 20 Kbytes EEPROM	20 Kbytes RAM	40 Kbytes RAM	40 Kbytes RAM; 40 Kbytes EEPROM	80 Kbytes RAM	
• expandable with memory card	-	up to 512 Kbytes	up to 512 Kbytes	-	up to 512 Kbytes (in CPU programmable up to 256 Kbytes)	
Process image size, inputs and outputs	32 bytes + 4 on-board	128 bytes	128 bytes	124 bytes + 4 on-board	128 bytes	
I/O address area						
• digital inputs/outputs	Inputs: 128 + 10 on-board Outputs: 128 + 6 on-board	128	512	Inputs: 496 + 20 on-board Outputs: 496 + 16 on-board	1024	
• analog inputs/outputs		32	64	Inputs: 64 + 4 on-board Outputs: 64 + 1 on-board	128	
Bit memory	1024			2048		
Counters	32			64		
Timers	64			128		
Max. sum of all retentive data	72 bytes		4736 bytes	144 bytes	4736 bytes	

Fontes e Chassis



Os chassis e as fontes de alimentação modulares fornecem flexibilidade na configuração de sistema. Selecionando o chassi, a fonte de alimentação, e os módulos apropriados do processador central e de E/S, você pode criar um sistema do controlador projetado especificamente para sua aplicação.

Quatro fontes de alimentação estão disponíveis para atender as exigências de potência do seu sistema; três fontes de alimentação de entrada CA e uma de entrada CC.

S7-300	BC	Descrição	MLFB
TRILHOS	TRILHO 160mm		6ES7390-1AB60-0AA0
	TRILHO 482mm		6ES7390-1AE80-0AA0
	TRILHO 530mm		6ES7390-1AF30-0AA0
	TRILHO 830mm		6ES7390-1AJ30-0AA0
	TRILHO 2000 mm		6ES7390-1BC00-0AA0
PS 307	Fonte de Alimentação - AC120/230V;DC24V;2A		6ES7307-1BA00-0AA0
	Fonte de Alimentação - AC120/230V;DC24V;5A		6ES7307-1EA00-0AA0
	Fonte de Alimentação - AC120/230V;DC24V;10A		6ES7307-1KA00-0AA0
PS 305	Fonte de Alimentação - 24DC/48/60/110V ; 24DCV ; 2A		6ES7305-1BA80-0AA0

MÓDULOS DE ENTRADAS E SAÍDAS

Módulos de Entradas e Saídas Digitais são:

S7-300	Descrição		MLFB
SM	Ent. Digital 16 x 24Vdc, isolada NEW!		6ES7321-1BH02-0AA0
DIGITAL	Ent. Digital 16 x 24Vdc, isolada, leitura Neg.		6ES7321-1BH50-0AA0
	Ent. Digital 32 x 24 Vdc		6ES7321-1BL00-0AA0
	Ent. Digital 16 x 120/230 Vac		6ES7321-1FH00-0AA0
	Ent. Digital 32 x 120 Vac		6ES7321-1EL00-0AA0
	Ent. Digital 8 x 120/230Vac		6ES7321-1FF01-0AA0
	Ent. Digital 8 x 120/230Vac; pontos isolados um a um		6ES7321-1FF10-0AA0
	Ent. Digital 16 x 24 Vdc, capac. interrupção e diagnose		6ES7321-7BH00-0AB0
	Ent. Digital 16 x 24 Vdc, p/ larga faixa de temp.		6ES7321-1BH82-0AA0
	Ent. Digital 16 x 48 - 125 Vdc p/ larga faixa de temp.		6ES7321-1CH80-0AA0
	Ent. Digital 32 x 24 Vdc p/ larga faixa de temperatura		6ES7321-1BL80-0AA0
	Saída Digital 8 x 24Vdc, 2A		6ES7322-1BF01-0AA0
	Saída Digital 16 x 24Vdc; 0,5 A		6ES7322-1BH01-0AA0
	Saída Digital 32 x 24 Vdc; 0,5 A		6ES7322-1BL00-0AA0
	Saída Digital 16 x 120 Vac; 0,5 A		6ES7322-1FH00-0AA0
	Saída Digital 32 x 120 Vac; 1 A; isolada (larg. dupla)		6ES7322-1EL00-0AA0
	Saída Digital 8 x 120 /230 Vac; 1 A		6ES7322-1FF01-0AA0
	Saída Digital 8 x Rele; 24Vdc/230Vac;2A		6ES7322-1HF01-0AA0
	Saída Digital 8 x Rele; 24Vdc/230Vac;5A	verif. conector	6ES7322-1HF10-0AA0
	Saída Digital 8 x Rele; 24Vdc/230Vac;5A p/ carga indutiva	verif. conector	6ES7322-5HF00-0AB0
	Saída Digital 16 x Rele; 24Vdc/230Vac; 2A		6ES7322-1HH01-0AA0
	Saída Digital 8 x 24 Vdc; 5A ou 230 Vac,5A para larga faixa de temp.		6ES7322-1HF80-0AA0
	Saída Digital 8 x 48-125V, 1.5A para larga faixa de temp.		6ES7322-1CF80-0AA0
	Saída Digital 8 x 120 Vac; 2 A; pontos isolados um a um		6ES7322-5FF00-0AB0

	Saída Digital 8 x 24Vdc; 0,5A; capac. diagnose		6ES7322-8BF00-0AB0
	Ent. Dig. 8 x 24Vdc + Saída Dig. 8 x 24 Vdc; 0,5A		6ES7323-1BH01-0AA0
	Ent. Dig. 16 x 24Vdc + Saída Dig. 16 x 24 Vdc; 0,5A		6ES7323-1BL00-0AA0

Módulos de Entradas e Saídas Analógicos

SM	Ent. Analógica 8 x 13 bits; I/U;PT100		6ES7331-1KF00-0AB0
ANALÓGICO	Ent. Analógica 8 x 12/14+S bits; I/U;PT100,Ni100;Thermo	4 Ent. p/ resist.	6ES7331-7KF02-0AB0
	Ent. Analógica 2 x 12/14+S bits; I/U; PT100; Ni100; Thermo	1 Ent. p/ resist.	6ES7331-7KB02-0AB0
	Ent. Analógica 2 x 12/14+S bits; I/U; PT100; Ni100; Thermo; para larga faixa de temp.	1 Ent. p/ resist.	6ES7331-7KB82-0AB0
	Ent. Analógica 8 x 16 bits; I/U; isolada		6ES7331-7NF00-0AB0
	Ent. Analógica 4/8 x 16 bits; I/U; isolada		6ES7331-7NF10-0AB0
	Ent. Analógica 2 x HART 0/4-20 mA	p/ IM153-2	6ES7331-7TB00-0AB0
	Ent. Analógica 8 x 16 bits , PT100/200/1000 NI100/120/200/500/ 1000		6ES7331-7PF00-0AB0
	Ent. Analógica 8 x 16 bits Thermopar Tipo B, E, J, K, L, N, R, S, T.		6ES7331-7PF10-0AB0
	Saída Analógica 2 x 12+S Bits; I/U		6ES7332-5HB01-0AB0
	Saída Analógica 2 x 11+S Bits; I/U; para larga faixa de temp.		6ES7332-5HB81-0AB0
	Saída Analógica 2 x 0-20 mA HART		6ES7332-5TB00-0AB0
	Saída Analógica 4 x 11+S Bits; I/U		6ES7332-5HD01-0AB0
	Saída Analógica 8 x		6ES7332-5HF00-

	11+S Bits; I/U	0AB0
	Saída Analógica 4 x 15 Bits; I/U; isolada	6ES7332-7ND00-0AB0
	Ent. Analog. 4x 8 Bits; I/U + Saída Analog. 2 x 8 Bits; U/I	6ES7334-0CE01-0AA0
	Ent. Analog. 4 x 12 Bits + Saída Analog. 2 x 12 Bits; 0-10V / PT100	6ES7334-0KE00-0AB0
	Ent + Sai Analog 4AI, 2AO, 12 bits 0-10V PT100	6ES7334-0KE80-0AB0
	Ent + Sai Analog 4I, 14 bits/4O, 11 ou 12 bits	6ES7335-7HG01-0AB0

EXERCÍCIO

Observe o código de cada placa do CLP e levante as características das placas a partir das tabelas das páginas anteriores.

- **Fonte**

Código de catálogo: _____

Tensão de entrada nominal: _____

Variação da tensão de entrada: _____

Potência de entrada: _____

Corrente de Saída: _____

Fusível: _____

Corrente da fonte do usuário: _____

- **Placa CPU**

Código de catálogo: _____

Memória do usuário: _____

Número de pontos de E/S: _____

Máximo de E/S analógicas (local): _____

Tempo de Varredura do Programa / Kpalavra: _____

Tempo de varredura de E/S: _____

Número Máximo de Chassis: _____

Número máximo de módulos de E/S (local): _____

Portas de Comunicação: _____

Memória de Backup: _____
Bateria para RAM: _____
Consumo de corrente da CPU: _____

• **Placa 1**

Código de catálogo: _____
Entrada ou saída: _____
Digital ou analógica: _____
Número de entradas/saídas: _____
Tipo AC ou DC: _____
Tensão de operação: _____
Aplicações: _____

• **Placa 2**

Código de catálogo: _____
Entrada ou saída: _____
Digital ou analógica: _____
Número de entradas/saídas: _____
Tensão de operação: _____
Corrente Máxima por Saída: _____
Corrente Máxima por Módulo: _____
Tipo de Contatos: _____
Aplicações: _____

• **Placa 3**

Código de catálogo: _____
Entrada ou saída: _____
Digital ou analógica: _____
Número de entradas/saídas: _____
Faixa de tensão/corrente de
operação: _____
Tempo de atualização: _____
Resolução máxima (bits): _____

* Pacotes de Configuração fornecidos juntamente com as placas.

** Necessário o uso de ferramentas de configuração (veja em SW)

*** Cabos RS232C. Ver Manuais e Acessórios

**** Pacote de configuração disponível com o firmware, verifique em "S7-300 / outros"

*****p/ CPU31X Nova Geração, CPU31XC, IM151

Endereçamento

Bits, Bytes e Palavras

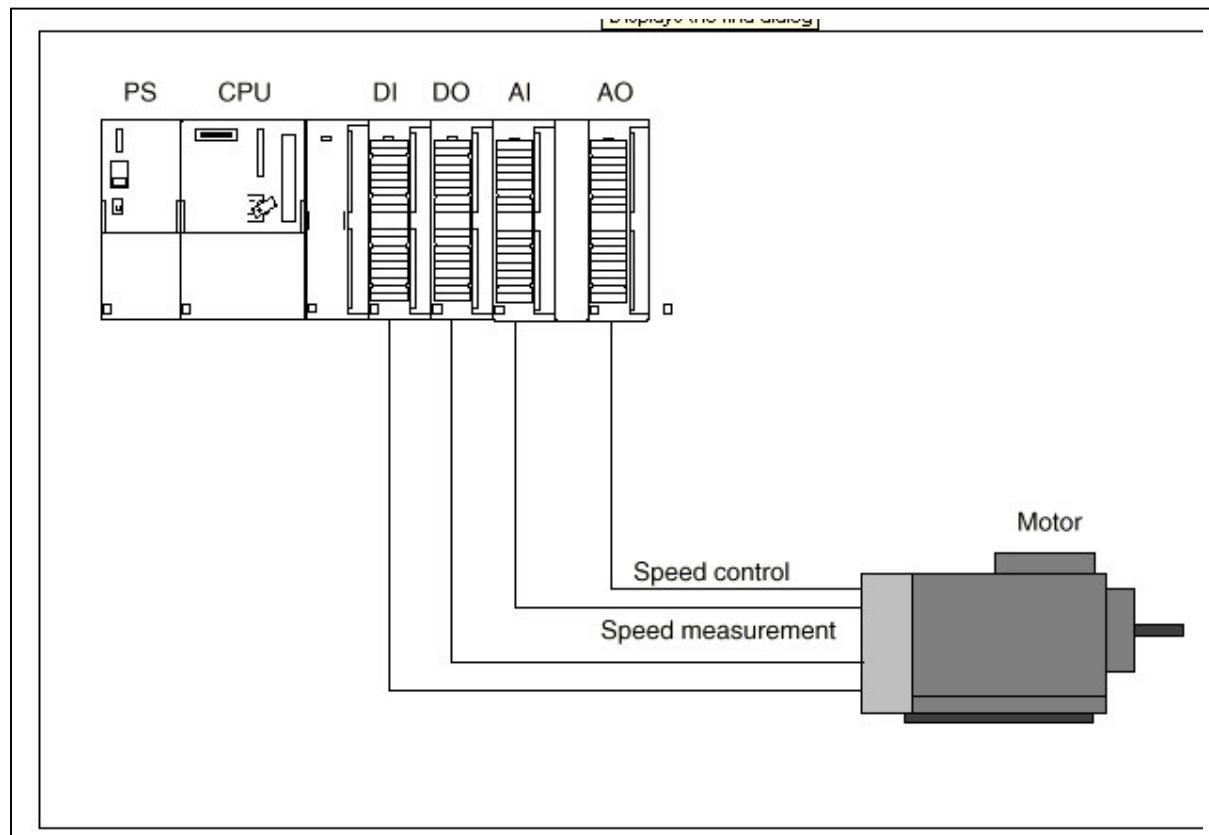
Bit: Dígito Binário (Binary digiT) – é a menor quantidade de informação possível. Tem dois estados: 0 e 1. 0 (zero) corresponde a desligado, desativado, off, aberto, apagado.

Byte: Conjunto de 8 bits, que corresponde a 1 caractere: 1 letra, 1 número ou 1 símbolo gráfico: < > , . : ? / { } [] + = - _ * & % \$ # @ !

Palavra: Número de bits que podem ser manipulados simultaneamente por um sistema. Na família SLC 500, a palavra é de 16 bits.

Numeração das Placas / Slots

Os Slots são numerados da esquerda para direita, contando a partir de um. O slot 1 é reservado para a fonte de alimentação, o slot dois para CPU e o 3 para o cartão de comunicação. Os outros Slots são usados para cartões de I/O. Caso haja mais de um rack, a numeração continua a partir do último slot do rack anterior. As placas assumem o número do slot onde estão encaixadas.



Endereçamento de Entradas e Saídas

Q125.0



Número da entrada ou saída
Número do Byte surgerido pelo programa S7.
Tipo: I = entrada
Q = Saída

Este exemplo mostra o endereço de uma saída digital (on-off), a saída 0 da Byte 125. Caso a saída (ou entrada) fosse analógica o endereço fica estabelecido da seguinte forma PQBX (endereço do Byte), PQWX (para dois Bytes, X é o endereço do primeiro Byte) e PQDX (para quatro Bytes, X é o endereço do primeiro Byte). O mesmo procedimento é feito para endereçamento de entrada física (I) e memória(M). Assim, substitui a letra "Q" pelas letras "I" e "M" quando estiver referenciando a entradas físicas e endereços de memória respectivamente.

Endereçamento de Bits e Palavras

Os endereços identificam áreas da memória RAM e são compostos de caracteres alfanuméricos separados por delimitadores. O ponto para acessar bits.

Exemplos:

M100.7 é um **endereço de elemento**, onde o **ponto** separa o endereço do Byte de memória (100) do oitavo bit.

MW100 endereça uma posição de memória de 16 **bits**, onde 100 equivale ao primeiro bytes da palavra (Word).

1.2.2. Constantes

Use esse método quando fornecer constantes para parâmetros da instrução.

Para fornecer uma constante hexadecimal: digite o valor hexadecimal seguido pela letra H (Hexadecimal).

Para fornecer uma constante binária: digite o valor binário seguido pela letra B (Binário). Por exemplo: digite 1010111101B, o mostrador exibe o equivalente hexadecimal (02BDh).

Para fornecer uma constante decimal: digite o valor decimal.

Diagrama de pinagem para a CPU S7-314 IFM

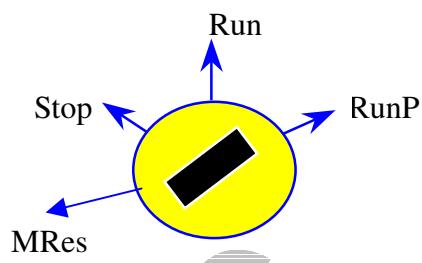
Integrated inputs/outputs				
	Special	Digital	IN OUT	
θ 1	I 126.0	θ 1	L+	2θ 1
θ 2		θ 2	124.0	2θ 2
θ 3	1	θ 3	1	2θ 3
θ 4	2	θ 4	2	2θ 4
θ 5	3	θ 5	3	2θ 5
θ 6	AO _U 128	θ 6	4	2θ 6
θ 7	AO _I 128	θ 7	5	2θ 7
θ 8	AI _U 128	θ 8	6	2θ 8
θ 9	AI _I 128	θ 9	7	2θ 9
1θ 0	AI- 128	1θ 0	M	3θ 0
1θ 1	AI _U 130	1θ 1	L+	3θ 1
1θ 2	AI _I 130	1θ 2	125.0	3θ 2
1θ 3	AI- 130	1θ 3	1	3θ 3
1θ 4	AI _U 132	1θ 4	2	3θ 4
1θ 5	AI _I 132	1θ 5	3	3θ 5
1θ 6	AI- 132	1θ 6	4	3θ 6
1θ 7	AI _U 134	1θ 7	5	3θ 7
1θ 8	AI _I 134	1θ 8	6	3θ 8
1θ 9	AI- 134	1θ 9	7	3θ 9
2θ 0	M _{ANA}	2θ 0	M	4θ 0

Diagrama de Pinagem da CPU 314 IFM

Modos de Operação

RumP	modo programação	Permite alteração no programa que está sendo executado pelo CLP.
Stop	modo remoto	Desabilita todas as Saídas. Não permite execução de programas residentes na CPU.
RUN	modo execução	Executa o programa. Não permite que o programa seja alterado. Habilita as Saídas.
MRES	Limpa as Mémorias	Limpa as memórias de usuários do CLP e faz o reset de falha (deve ser pressionado durante 3 segundos aproximadamente).

O modo de operação é selecionado pela chave rotativa que fica na frente da CPU.



2.6 Linguagens de Programação de CLPs

A norma IEC 1131 visa normatizar os controladores lógicos programáveis (CLP).

Essa norma é composta de 5 partes:

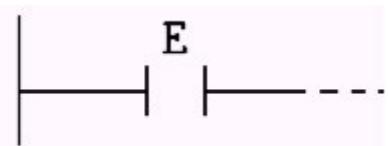
1. Informações gerais e características dos CLPs.
2. Equipamentos elétricos e mecânicos requeridos.
3. Linguagens de programação de CLPs.
4. Guia de auxílio ao usuário no projeto de sistemas de automação.
5. Comunicação entre CLPs de diferentes fabricantes.

Neste material são mostrados os aspectos básicos do diagrama de contatos, ou diagrama escada (mais comumente chamado de diagrama Ladder), por ser a linguagem de programação de CLPs mais difundida no meio técnico. É mostrada também a linguagem STL, de propriedade da SIEMENS, que, apesar de não seguir a norma, é a linguagem implementada que mais se aproxima do IL, normatizada pela IEC 1131-3 (IEC, 1992). O STL foi escolhido para ser utilizado neste trabalho por ser uma linguagem texto, permitindo, assim, a geração automática do programa para o compilador do CLP a partir de uma ferramenta computacional. Já o Ladder, apesar de não permitir tal geração, segue a padronização da norma IEC 1131-3.

3. Introdução a Linguagem Siemens

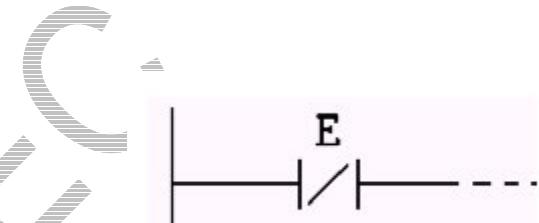
3.1 Ladder

Um degrau do diagrama Ladder (SIEMENS 1966a) é composto por duas partes: a primeira, sempre à esquerda, representa uma expressão lógica. A segunda, à direita, pode representar um sinal de saída, uma memória ou um comando especial. Os sinais de entrada e a leitura dos flags internos são representados por duas barras paralelas, indicando contato elétrico caso o sinal seja verdadeiro, ver figura abaixo.



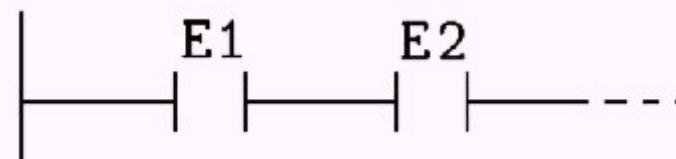
Sinal de entrada no diagrama Ladder.

Duas barras paralelas cortadas em diagonal representam a lógica inversa do sinal



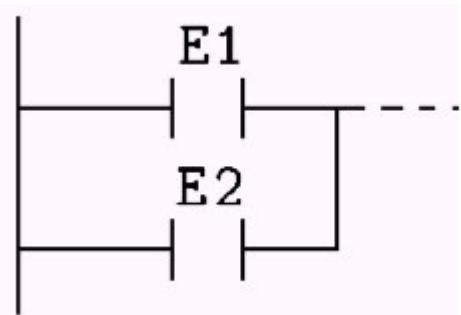
Sinal de entrada inversa no diagrama Ladder.

A lógica **e** é representada por dois sinais em série, ou seja, caso um dos sinais seja falso, o circuito é aberto, tornando falso o sinal de saída. A figura abaixo mostra a lógica **e**.



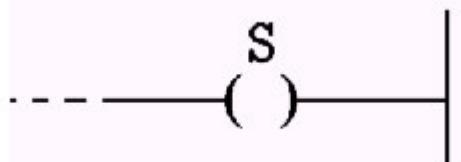
Lógica **e** no diagrama Ladder.

A lógica **ou** é representada por dois sinais em paralelo. Caso um dos sinais seja verdadeiro, o circuito é fechado, tornando verdadeiro o sinal de saída. A figura abaixo mostra a lógica **ou**.



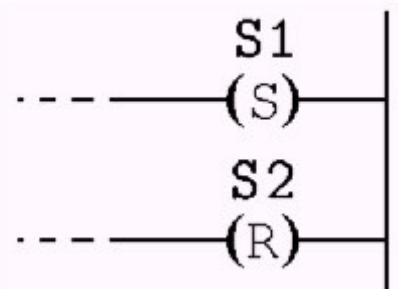
Lógica **ou** no diagrama Ladder.

Os sinais de saída e a escrita nos `\textit{flags}` internos são representados por dois parênteses na parte direita do degrau. Caso haja sinal no circuito (circuito fechado, ou lógica de entrada verdadeira) é escrito o valor verdadeiro na saída, caso contrário é escrito o valor falso. Nesse caso, sempre será escrito algum valor na variável de saída.



Atribuição de valor a um sinal de saída em diagrama Ladder.

Pode ser desejado, porém, alterar o valor da saída apenas se a lógica de entrada for verdadeira. Nesse caso, são utilizadas instruções de set e reset. A instrução de set escreve o valor verdadeiro na saída se a lógica de entrada for verdadeira. A instrução de reset escreve o valor falso na saída se a lógica de entrada for verdadeira. A figura abaixo mostra a representação das duas instruções.



Instruções de set e reset

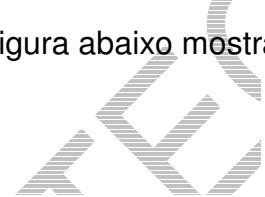
A instrução NOT inverte o valor lógico em um ponto do circuito. A figura abaixo mostra a lógica

$$S \leftarrow \overline{B \otimes C}.$$

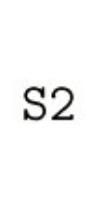
Instrução NOT em diagrama Ladder.}

Variáveis ou instruções de saída diferentes com lógicas de entrada iguais podem ser agrupadas em paralelo em um mesmo degrau.

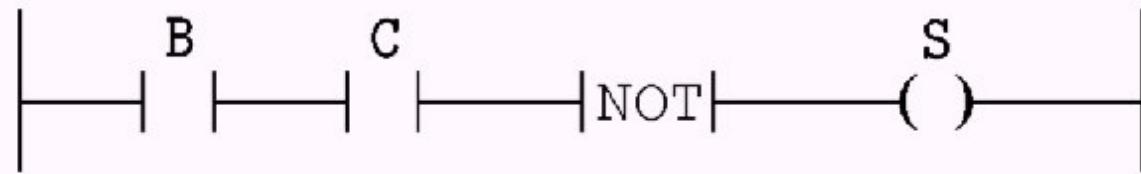
A figura abaixo mostra a representação da lógica:



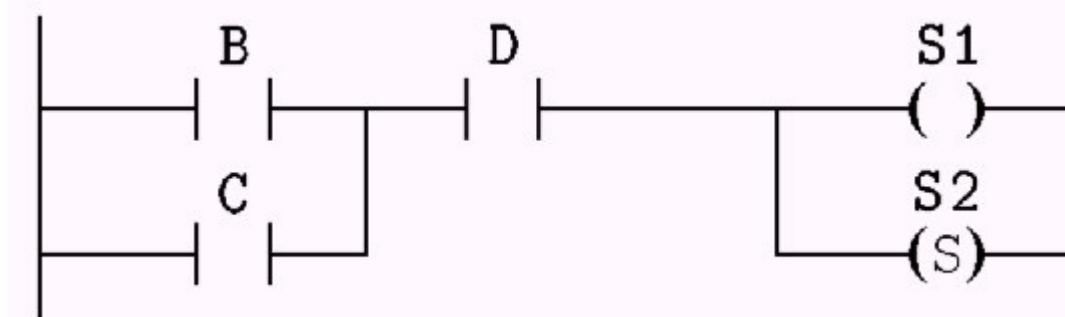
$$S1 \leftarrow (B \oplus C) \otimes D$$



$$S2 \leftarrow \begin{cases} \text{verdadeiro} & \text{se } (B \oplus C) \otimes D = \text{verdadeiro}, \\ S2 & \text{se } (B \oplus C) \otimes D = \text{falso}. \end{cases}$$



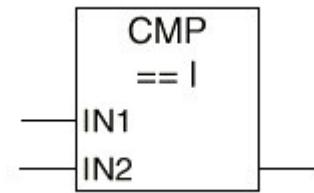
Representação da instrução NOT em LADER da equação acima.



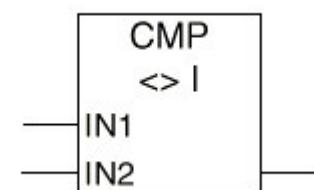
Exemplo - representação em Ladder da lógica da equação

Funções especiais de comparação variam de fabricante para fabricante, porém são sempre utilizadas na parte esquerda do degrau Ladder. Na figura tem-se a

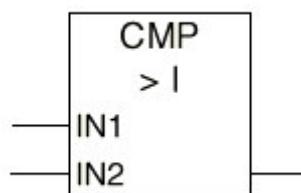
comparação de dois inteiros. Caso o resultado seja verdadeiro, o circuito é fechado, caso contrário o circuito é aberto.



IN1 Igual IN2



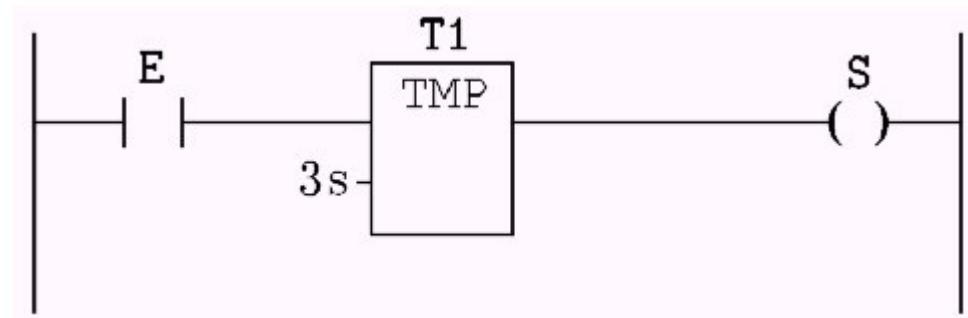
IN1 Diferente IN2



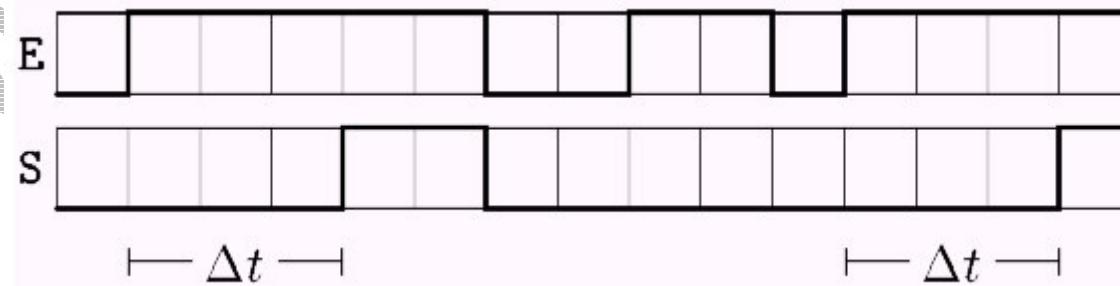
IN1>IN2

Função de comparação de inteiros no diagrama Ladder.

Da mesma maneira, os temporizadores variam conforme o fabricante, mas também são utilizados na parte esquerda do degrau. Porém, por possuirem uma entrada de habilitação, nunca são o primeiro elemento do degrau, participando sempre de uma lógica \textbf{e}. Existem vários tipos de temporizadores, dos quais o mais utilizado é o temporizador com retardo na ligação. Quando o sinal aplicado na sua entrada é verdadeiro, inicia-se a contagem do tempo. No momento em que essa contagem atinge o valor pré-determinado, o circuito é fechado e o sinal de saída passa a ser verdadeiro, só voltando a ser falso quando o sinal de entrada se torna falso. A figura abaixo mostra a representação de um temporizador de 3 segundos em diagrama Ladder, tendo como sinal de habilitação a entrada E e como saída S. Logo após ao diagrama LADDER mostramos também o gráfico relacionando os sinais de entrada e de saída para esse temporizador.

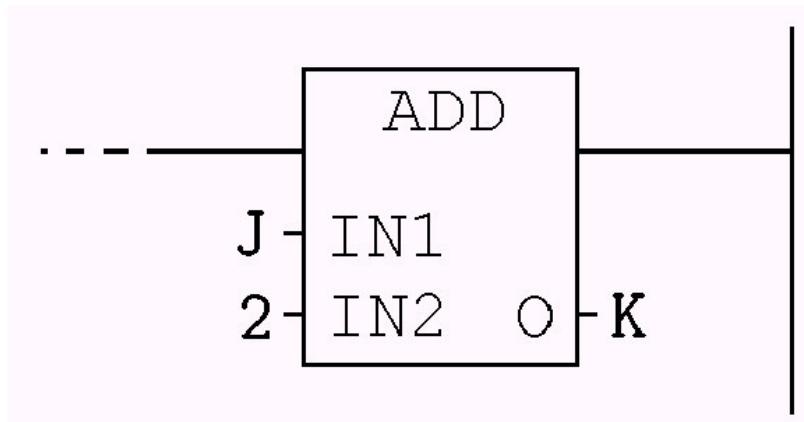


Temporizador no diagrama Ladder.



Entrada X saída para um temporizador com retardo na ligação.

Operações matemáticas podem ser realizadas utilizando instruções como ADD (adição) e SUB (subtração). Essas instruções são executadas somente quando o sinal em sua entrada de habilitação for verdadeiro, e são utilizadas na parte direita do degrau. Devem ser fornecidos dois valores constantes ou variáveis sobre os quais será realizada a operação, e uma variável na qual será armazenado o resultado. A instrução ADD na figura abaixo mostra a soma da variável J com a Constante 2, sendo o valor armazenado em K.



Instrução de soma em diagrama Ladder.

3.2 STL

O STL-Statement List é uma linguagem texto para programação de CLPs, desenvolvida pela SIEMENS. Essa linguagem é próxima da IL- Instruction List, padronizada pela norma IEC 1131-3, se diferenciando basicamente pela sintaxe dos comandos. Segundo o próprio fabricante, devido à grande difusão da linguagem STL e à posição de liderança no mercado de CLPs, foi dada, no desenvolvimento das versões recentes, prioridade à compatibilidade com versões anteriores do próprio STL. Na verdade, não existe ainda implementada nenhuma linguagem de programação totalmente equivalente à IL.

Aqui é apresentada a sintaxe utilizada pela linguagem STL por ter sido utilizada para validação do procedimento.

Em STL, o resultado de uma operação lógica (Result of Logic Operation) é armazenado em um flag interno chamado RLO. Toda operação subsequente é realizada com o RLO. Por exemplo, a operação A B (AND B) é equivalente a

$$\text{RLO} \leftarrow \text{RLO} \otimes \text{B.}$$

Comparando-se com o diagrama Ladder, o RLO é equivalente à tensão nos contatos após cada elemento do circuito.

O RLO pode, então, ativar ou desativar um sinal de saída, ou servir de sinal de habilitação para uma instrução como um temporizador.

O termo first check (primeira contagem) indica que está sendo executada a primeira instrução de uma lógica. Isso significa que uma nova operação lógica se iniciou, e que o resultado (RLO) da operação lógica anterior não será executado. Isso torna sem importância qual instrução (por exemplo, **e** ou **ou**) está sendo executada como primeira instrução de uma lógica escrita em STL.

As operações **e** e **ou** são feitas, respectivamente, pelas instruções A e 0, que têm como argumento o nome da variável. O valor do RLO é armazenado em uma variável lógica (saída ou flag interno) através da instrução "**=**". A figura abaixo mostra como pode ser escrita em STL a operação

$$S \leftarrow (B \oplus C) \otimes D$$

Note que a primeira instrução tanto pode ser 0 B quanto AB.

0	B
0	C
A	D
=	S

Instruções ou e e em STL.

As instruções "AN" e "ON" fazem, respectivamente, as operações e e ou com os valores inversos de seus argumentos. A figura abaixo mostra a operação

$$S \leftarrow (B \oplus \bar{C}) \otimes D.$$

Instruções **ou**(valor inverso) e **e** em STL.

0	B
ON	C
A	D
=	S

Instruções **ou**(inverso) e **e** em STL.

As instruções de set e reset são representadas pelas letras "S" e "R". A figura abaixo mostra essas instruções.

A	B
S	S1
A	C
R	S2

Instruções de set e reset em STL.

Assim como em Ladder, variáveis ou instruções de saída diferentes com lógicas de entrada iguais podem ser agrupadas em um mesmo bloco, utilizando o mesmo valor do RLO (veja exemplo).

0	B
0	C
A	D
=	S1
S	S2

Representação em STL a lógica da equação abaixo.

$$S1 \leftarrow (B \oplus C) \otimes D$$

$$S2 \leftarrow \begin{cases} \text{verdadeiro} & \text{se } (B \oplus C) \otimes D = \text{verdadeiro}, \\ S2 & \text{se } (B \oplus C) \otimes D = \text{falso}. \end{cases}$$

A figura mostra a representação da lógica da equação, mostrada em Ladder no exemplo acima

A instrução "NOT" inverte o valor do RLO. A figura abaixo mostra a lógica

$$S \leftarrow \overline{B \otimes C}.$$

Em instruções de comparação, é utilizado o conceito de pilha, em que os valores a serem comparados são armazenados na pilha através da instrução "L" e, em seguida comparados através da instrução apropriada. Na figura abaixo compararam-se dois inteiros e armazena-se o resultado lógico da operação na saída "S".

A	B
A	C
NOT	
=	S

Instrução NOT em STL

L	J
L	2
>=I	
=	S

Função de comparação de inteiros em STL.

Nos casos em que o resultado da comparação deve passar por uma operação lógica, é utilizado o conceito de pilha de RLOs. A abertura de um parêntese após instruções de operação de bits (ex. A(), O(), AN(), ON() etc) grava o valor atual do RLO na pilha, inicia um novo RLO, executa as instruções seguintes e, quando o parênteses é fechado, é executada a operação lógica entre o RLO atual e o RLO da pilha. (Veja o exemplo).

A figura mostra a representação da lógica da equação, em que o flag "F" é ativado se a entrada "E" for verdadeira e o inteiro "J" for maior ou igual a "2".

$$F \leftarrow \begin{cases} \text{verdadeiro} & \text{se } (E = \text{verdadeiro}) \text{ e } (J \geq 2), \\ F & \text{caso contrário.} \end{cases}$$

A	E
A(
L	J
L	2
>=I	
)	
S	F

Exemplo - representação em STL da lógica da equação acima

Cada tipo de temporizador possui uma instrução diferente, sendo "SD" a instrução do temporizador com retardo na ligação. Essa instrução continua (ou inicia) a contagem de tempo se o RLO for verdadeiro. Antes da instrução "SD" é necessário armazenar na pilha (instrução "L") o tempo associado ao temporizador. A figura mostra a representação de um temporizador de 3 segundos em STL, tendo como sinal de habilitação a entrada "E" e como saída "S". Observe-se que o valor do tempo obedece ao formato ``"S5T#"'' seguido do valor propriamente dito.

A	E
L	S5T#3s
SD	T1
A	T1 = S

Operações matemáticas como soma e subtração são sempre executadas entre os dois últimos valores armazenados na pilha. O resultado da operação é armazenado na pilha, e pode ser transferido para uma variável através da instrução "T". A figura mostra a soma da variável "J" com a constante "2", sendo o valor armazenado em "K".

L	J
L	2
+I	
T	K

Instrução de soma em STL.}

Pode-se desejar executar uma operação matemática apenas se uma dada condição lógica for satisfeita. Como as instruções "L" e "T" não são influenciadas pelo valor do RLO, é necessário se utilizar uma instrução de salto condicional ("JNB" – saltar se RLO=falso) que provoca o desvio do fluxo do programa para a instrução seguinte à instrução \verb|verb| "T", que deve ser identificada por um {label}. Caso essa seja a última instrução de um bloco, salta-se para a instrução "NOP 0" (no operation) que deve ser inserida após a instrução "T". A figura mostra um programa STL que, caso o valor da entrada "E" seja verdadeiro, soma a variável "J" com a constante "2" e armazena o resultado em "K".

A	E
JNB	_001
L	J
L	2
+I	
T	K
_001: NOP 0	

Soma condicional em STL.

3.3 Tradução entre as linguagens

Todo programa escrito em Ladder pode ser traduzido para STL. Contudo, por ser mais flexível, nem sempre é possível se traduzir um programa de STL para Ladder.

Quando um programa STL pode ser traduzido para Ladder e é escrito em uma certa formatação, o ambiente de programação STEP7 da SIEMENS é capaz de fazer a conversão automaticamente. Porém isso geralmente torna o programa maior e de execução mais lenta.

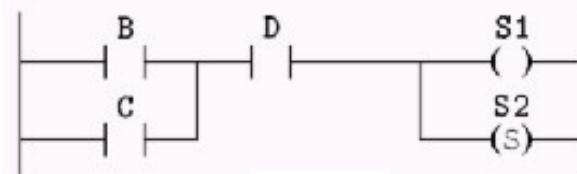
São mostrados, a seguir, alguns exemplos de programas em Ladder e sua tradução para STL na forma otimizada e na forma que permita sua conversão para Ladder pelo STEP7.

A figura abaixo mostra os programas para a lógica da equação

$$S1 \leftarrow (B \oplus C) \otimes D$$

$$S2 \leftarrow \begin{cases} verdadeiro & \text{se } (B \oplus C) \otimes D = \text{verdadeiro}, \\ S2 & \text{se } (B \oplus C) \otimes D = \text{falso}. \end{cases}$$

Equação



Ladder

0 B	0 B
0 C	0 C
A D	A D
= S1	= L0.0
S S2	A L0.0
	BLD 102
	S S2

STL otimizado

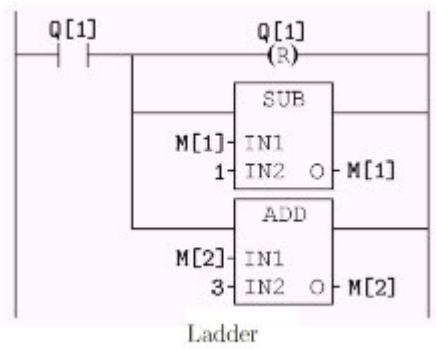
STL conversível

para Ladder

Programas em Ladder e STL

O resultado da operação é temporariamente armazenado no “L0.0”, que é utilizado em seguida para definir o valor de “S1” e “S2”. A instrução “BLD 102” não

tem função alguma, servindo simplesmente para possibilitar a tradução do programa para Ladder.



Ladder

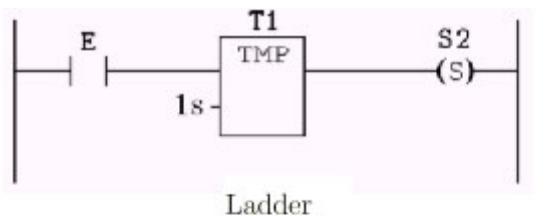
A	Q[1]	A	Q[1]
R	Q[1]	=	L0.0
JNB	_001	A	L0.0
L	M[1]	BLD	102
L	1	R	Q[1]
-I		A	L0.0
T	M[1]	JNB	_001
		L	M[1]
		L	1
		-I	
		T	M[1]
		_001:	NOP 0
		A	L0.0
		JNB	_002
L	M[2]	L	M[2]
L	3	L	3
+I		+I	
T	M[2]	T	M[2]
_001:	NOP 0	_002:	NOP 0

STL otimizado

STL conversível

para Ladder

- Tradução de programa de Ladder para STL.



Ladder

A	E
L	S5T#1s
SD	T1
A	T1
S	S2

STL otimizado

A	E
L	S5T#1s
SD	T1
A	T1
NOP	0
NOP	0
NOP	0
S	S2

STL conversível

para Ladder

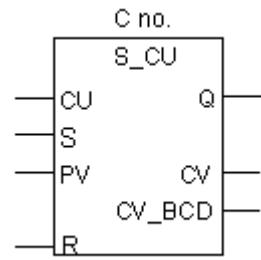
- Tradução de programa de Ladder para STL.

Para que o STEP7 traduza o programa para Ladder, necessário que cada bloco de operações matemáticas seja identificado por um *label*.

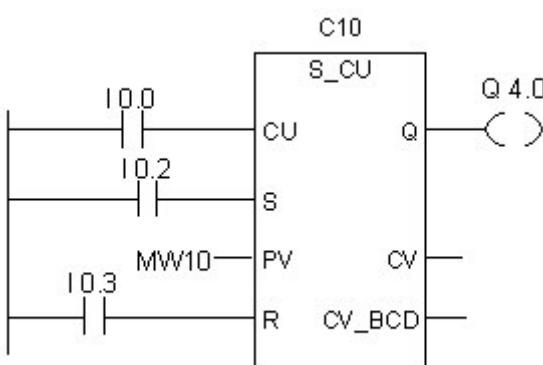
4. Instruções avançadas da Siemens

4.1 Contador Crescente S_CU

Com um “flanco de impulso” positivo na entrada S, o contador é setado com o valor da entrada PV. Iniciando com 0 ou PV, o contador conta crescentemente a cada vez que existe um flanco de impulso positivo na entrada CU. A saída Q é sempre 1, enquanto o valor de CV não for igual a 0. se houver um flanco de impulso positivo na entrada R o contador é resetado, isto é, o contador é setado com o valor 0.

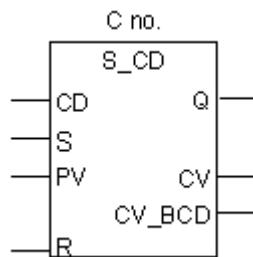


Exemplo de um contador S_CU

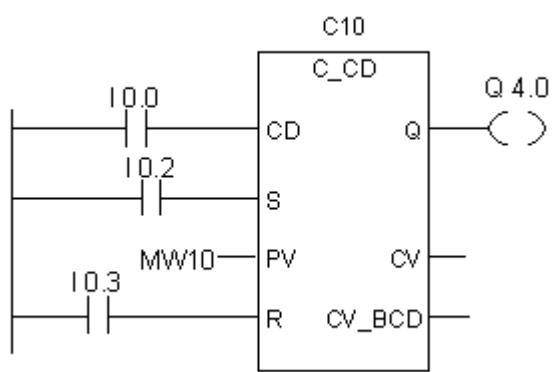


Contador decrescente S_CD

Com um “flanco de impulso” positivo na entrada S, o contador é setado com o valor da entrada SC. Iniciando com 0 ou SC, o contador conta decrescentemente a cada vez que existir um flanco de impulso positivo na entrada CD. A saída Q é sempre 1, enquanto o valor CV não for igual a 0. Se houver um flanco de impulso positivo na entrada R o contador é resetado, isto é, o contador é setado com o valor 0.

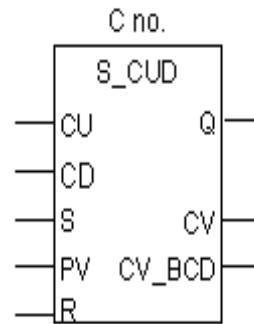


Exemplo de contador S_CD

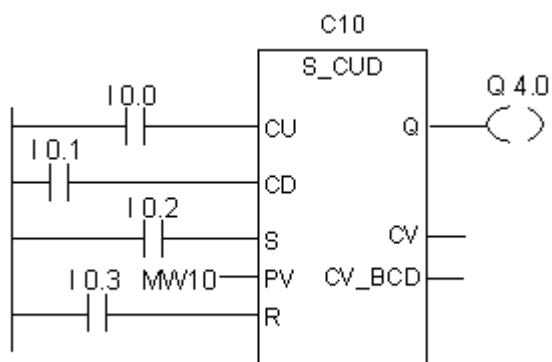


Up / Down Counter S_CUD

Combição de contadores crescente e decrescente



Exemplo de contador S_CUD



4.2 Instruções Matemáticas

4.2.1 Exemplos de Casos para PLCs Allen-Bradley

As Instruções Matemáticas consideram um par de valores e realizam a operação desejada. O resultado é colocado em uma localização separada.

Parâmetros das Instruções

Origem – Endereços dos valores em que a operação matemática será executada. Podem ser endereços de palavra ou constantes de programa. Se a instrução tiver dois operandos Origem, não é possível introduzir constantes de programa nos dois operandos.

Dest – Endereço destino referente ao resultado da operação.

Adição (ADD)

Adiciona o valor **Origem A** ao valor **Origem B** e armazena o resultado no destino.

A figura a seguir apresenta o formato da instrução

ADD	
Adicionar	
Origem A	C5:5.ACC
Origem B	35
Dest	N7:33

Subtração (SUB)

A instrução **SUB** subtrai o valor **Origem B** do valor **Origem A** e armazena o resultado no destino (Dest).

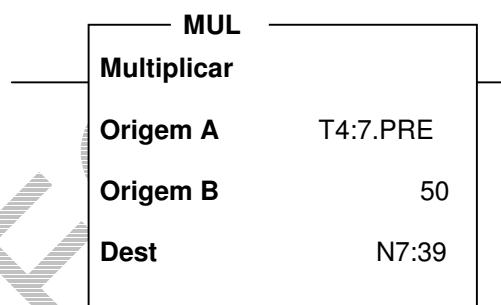
A figura a seguir apresenta o formato da instrução

SUB	
Subtrair	
Origem A	N7:10
Origem B	19
Dest	N7:33

Multiplicação (MUL)

A instrução **MUL** multiplica o valor origem A pelo valor origem B e armazena o resultado no destino (Dest).

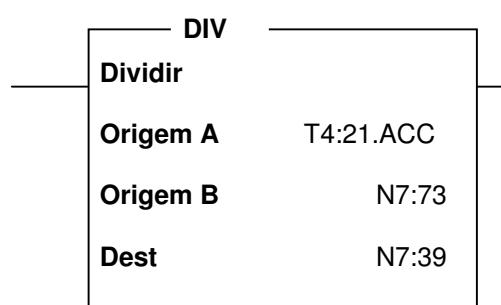
A figura a seguir apresenta o formato da instrução



Divisão (DIV)

A instrução **DIV** divide o valor origem A pelo valor origem B e armazena o resultado arredondado no destino (Dest).

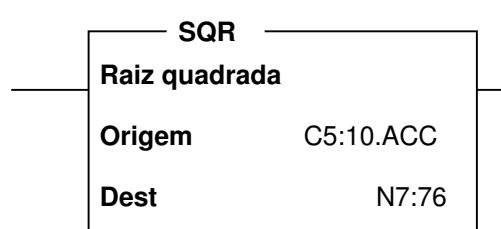
A figura a seguir apresenta o formato da instrução



Raiz Quadrada (SQR)

Calcula a raiz quadrada do valor Origem e coloca o inteiro resultante no destino Dest.

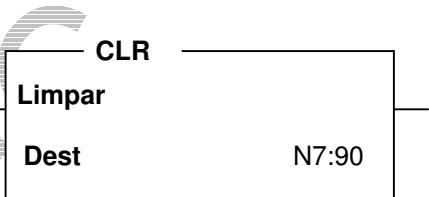
A figura a seguir apresenta o formato da instrução



Zeramento (CLR)

Zera todos os bits de uma palavra

A figura a seguir apresenta o formato da instrução



4.2.2 Estudo de Casos para PLCs Siemens

EN = Habilita entrada. A instrução será executada se e somente se o RLO é verdadeiro (RLO=1).

ENO = Habilita saída. A saída Enable output tem o mesmo estado de sinal que EN (EN=ENO), a menos que tenha havido um erro durante a conversão. Por exemplo, a instrução DIV_I fornece ENO=0 quando se faz uma divisão por zero.

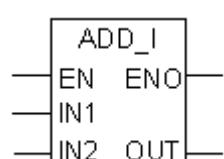
IN1 = Entrada 1 1. valor aritmético da instrução

IN2 = Entrada 2 2. valor aritmético da instrução

OUT = Saída - Resultado da operação aritmética

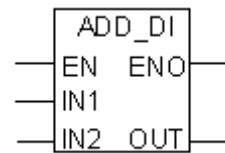
Adição

ADD_I Soma inteiros



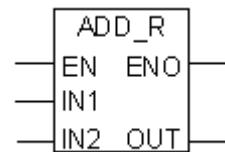
Adição de Inteiros

ADD_DI Soma inteiros duplos



Adição de inteiros duplos

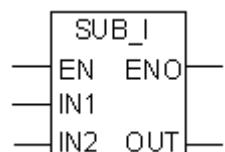
ADD_R Soam números reais



Adição de Reais

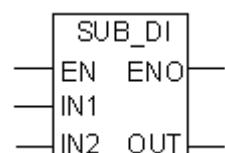
Subtração

SUB_I Subtrai inteiros



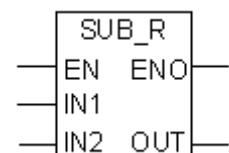
Subtração de Inteiros

SUB_DI Subtrai inteiros duplos



Subtração de Inteiros duplos

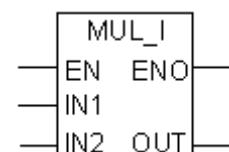
SUB_R Subtrai números reais



Subtração de Reais

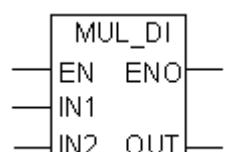
Multiplicação

MUL_I Multiplica Inteiros



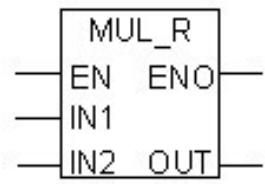
Multiplicação de Inteiros

MUL_DI Multiplica inteiros duplos



Multiplicação de Reais Duplos

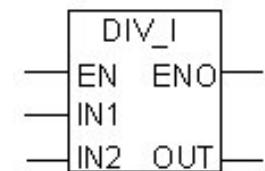
MUL_R Multiplica números reais



Multiplicação de Reais

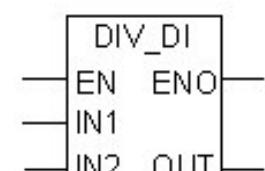
Divisão

DIV_I Divide inteiros



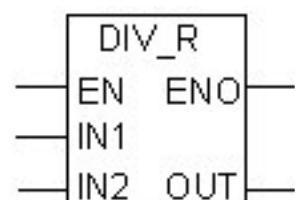
Divisão de Inteiros

DIV_DI Divide inteiros duplos



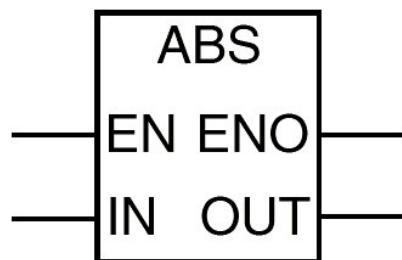
Divisão de Duplo Inteiros

DIV_R Divide números reais



Divisão de Reais

ABS_R Quando a entrada EN é verdadeira, o processador pega o valor da posição de memória IN e extrai o seu valor absoluto e armazena na posição OUT. A saída ENO reproduz a mesma condição existente em EN.



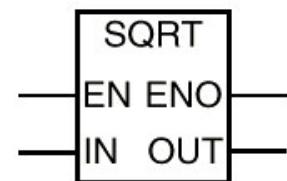
Valor Absoluto

SQR_R Eleva o valor armazenado em IN ao quadrado e guarda na posição apontada por OUT.



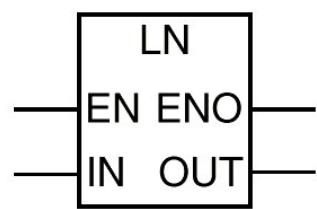
O quadrado de um Número

SQRT_R Extrai a raiz quadrada do valor armazenado em IN e guarda na posição apontada por OUT.



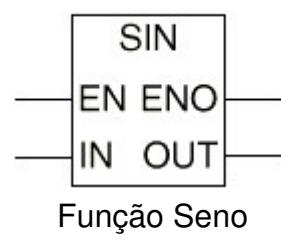
Raiz Quadrada.

LN_R Determina o logaritmo Neperiano do valor armazenado na posição de memória apontada em IN e salva na posição de memória apontada por OUT.



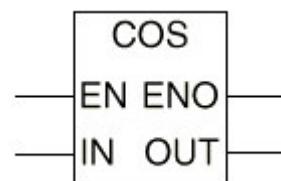
Logaritmo Neperiano.

EXP_R Calcula a exponencial de Neper do valor armazenado na posição de memória apontada por IN e salva o resultado na posição apontada por OUT.



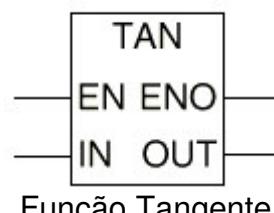
Função Seno

SIN_R Calcula o seno do valor armazenado na posição de memória apontada por IN e salva o resultado na posição apontada por OUT.



Função Cosseno

COS_R Calcula o Cosseno do valor armazenado na posição de memória apontada por IN e salva o resultado na posição apontada por OUT.



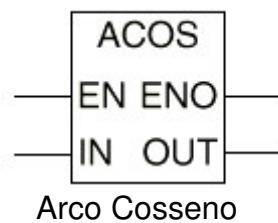
Função Tangente

TAN_R Calcula a Tangente do valor armazenado na posição de memória apontada por IN e salva o resultado na posição apontada por OUT.

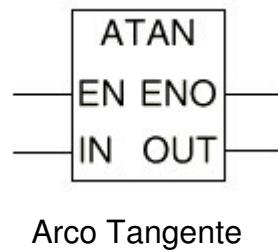


Arco Seno

ASIN_R Calcula o arco cujo seno é o valor armazenado na posição de memória apontada por IN e salva o resultado na posição apontada por OUT.



ACOS_R Calcula o arco cujo cosseno é o valor armazenado na posição de memória apontada por IN e salva o resultado na posição apontada por OUT.



ATAN_R Calcula o arco cujo a tangente é o valor armazenado na posição de memória apontada por IN e salva o resultado na posição apontada por OUT.

4.3 Instruções de Comparação

4.3.1 Estudo de Casos para PLCs da Allen-Bradley

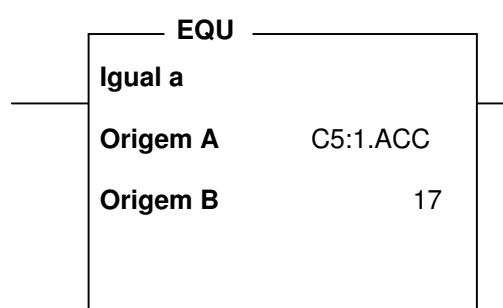
As instruções de Comparação são usadas para testar pares de valores de forma a condicionar a continuidade lógica de uma linha.

Igual a (EQU)

Testa se dois valores são iguais. Se a **Origem A** e **Origem B** são iguais, a lógica da linha é verdadeira.

Origem A deve ser um endereço. **Origem B** pode ser uma constante do programa ou um endereço.

A figura a seguir apresenta o formato da instrução

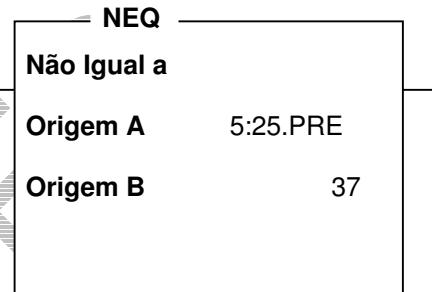


Diferente (NEQ)

Testa se o primeiro valor não é igual ao segundo. Se Origem A e Origem B são diferentes, a lógica da linha é verdadeira.

Origem A deve ser um endereço. **Origem B** pode ser uma constante do programa ou um endereço.

A figura a seguir apresenta o formato da instrução NEQ

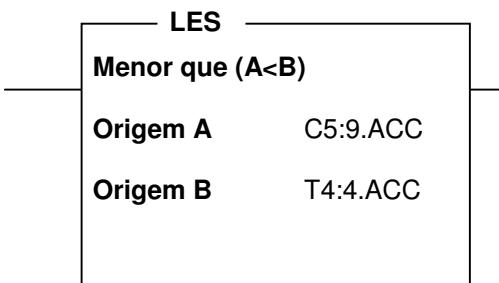


Menor que (LES)

Testa se o primeiro valor é menor que o segundo. Se a Origem A é menor que o valor da Origem B a lógica da linha é verdadeira.

Origem A deve ser um endereço. **Origem B** pode ser uma constante do programa ou um endereço.

A figura a seguir apresenta o formato da instrução

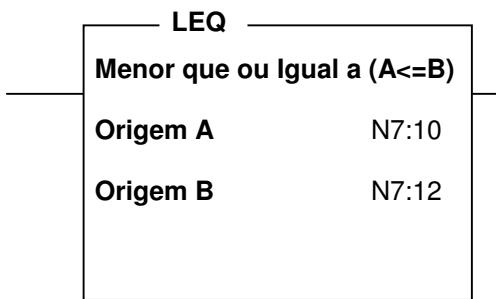


Menor ou igual a (LEQ)

Testa se o primeiro valor é menor ou igual ao segundo. Se o valor da Origem A é menor ou igual Origem B, a lógica da linha é verdadeira.

Origem A deve ser um endereço. **Origem B** pode ser uma constante do programa ou um endereço.

A figura a seguir apresenta o formato da instrução

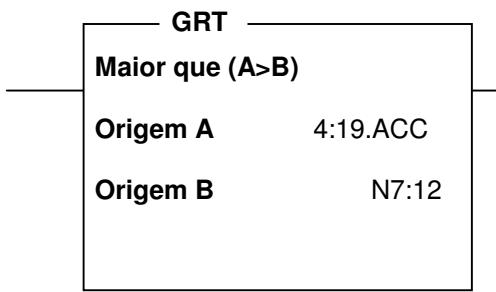


Maior que (GRT)

Testa se o primeiro valor é maior que o segundo. Se o valor da Origem A é maior que o valor da Origem B, a lógica da linha é verdadeira.

Origem A deve ser um endereço. **Origem B** pode ser uma constante do programa ou um endereço.

A figura a seguir apresenta o formato da instrução

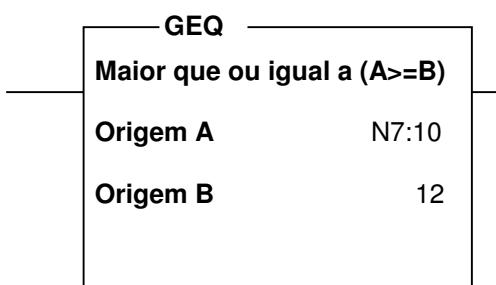


Maior ou igual a (GEQ)

Testa se o primeiro valor é maior ou igual ao segundo. Se o valor da Origem A é maior ou igual ao valor da Origem B, a lógica da linha é verdadeira.

Origem A deve ser um endereço. **Origem B** pode ser uma constante do programa ou um endereço.

A figura a seguir apresenta o formato da instrução



Teste de Limite (LIM)

Exemplo:

LIM	
Teste de Limite	
Lim Inf	590
Teste	N7:12
Lim Sup	610

LIM testa se o valor de **Teste** está dentro ou fora da faixa especificada por **Limite Inferior** (Lim Inf) e **Limite Superior** (Lim Sup).

Para testar se o valor de **Teste** está dentro da faixa, o **Limite Inferior** deve ter um valor igual a ou menor que o **Limite Superior**. A instrução será verdadeira quando o valor de Teste estiver entre os limites ou for igual a um dos limites. Se o valor de Teste estiver fora dos limites, a instrução será falsa.

Para testar se o valor de **Teste** está fora da faixa, o **Limite Inferior** deve ter um valor maior que o **Limite Superior**. A instrução será verdadeira quando o valor de Teste estiver fora dos limites ou for igual a um dos limites. Se o valor de Teste estiver entre os limites, a instrução será falsa.

Fornecendo Parâmetros

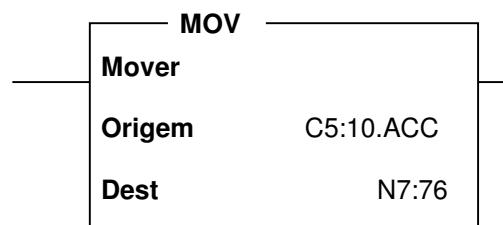
Dependendo de como você define o parâmetro **Teste**, os parâmetros **Limite Inferior** e **Limite Superior** podem ser um endereço de palavra ou uma constante de programa. Veja abaixo.

Se Teste for	Limite Inferior	Limite Superior
Constante	Endereço de Palavra	Endereço de Palavra
Endereço de Palavra	Endereço de Palavra ou Constante	Endereço de Palavra ou Constante

Mover (MOV)

Move o valor da origem para o destino;

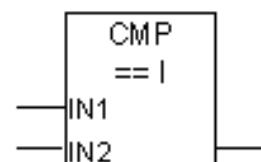
A figura a seguir apresenta o formato da instrução



4.3.2 Estudos de Casos para o CLP Siemens

Para o PLC SIEMENS, a grande maioria das instruções apresentam-se vinculadas ao tipo de variável que será manipulada.

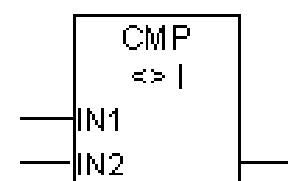
Igual a(I para inteira, D para inteiro duplo e R para reais)



Símbolo para igualdade de Inteiros

A instrução de comparação “igual a” habilita a saída se IN1 for igual a IN2

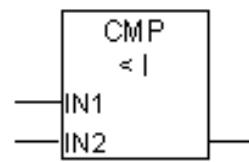
Diferente a(I para inteira, D para inteiro duplo e R para reais)



Símbolo para diferença entre inteiros

A instrução de comparação “não igual a” habilita a saída se IN1 for diferente de IN2

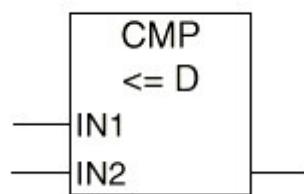
Menor que(I para inteira, D para inteiro duplo e R para reais)



Símbolo para comparação entre inteiros

A instrução “menor que” habilita a saída se $IN1 < IN2$

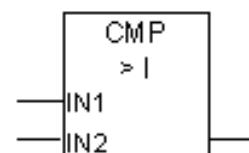
Menor ou igual a(I para inteira, D para inteiro duplo e R para reais)



Símbolo para comparação entre Duplo inteiros

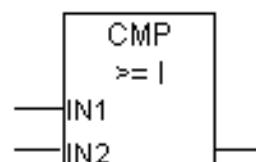
A instrução “menor ou igual a” habilita a saída se $IN1 \leq IN2$

Maior que(I para inteira, D para inteiro duplo e R para reais)



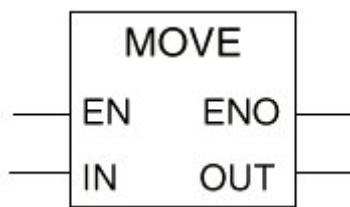
A instrução “maior que” habilita a saída $IN1 > IN2$

Maior ou igual a(I para inteira, D para inteiro duplo e R para reais)



A instrução “maior ou igual a” habilita a saída $IN1 \geq IN2$

MOVE _ A instrução MOVE movimenta um Byte(8 bits), Word(16 bits) ou DWord(32 bits) da posição apontada por IN para uma outra apontada por OUT. Isto ocorre quando a entrada EN se encontra habilitada.



Símbolo da Instrução MOVE

4.4 Instruções de Controle de Programa

4.4.1 Para CLP ALLEN-Bradley

JMP [Saltar para Rótulo] e LBL [Rótulo]

Exemplo :

_____(JMP)_____
_____**1 LBL**_____

JMP faz o processador saltar à frente ou atrás para a instrução de rótulo (LBL) correspondente e retomar a execução do programa a partir do rótulo.

LBL é o alvo da instrução JMP com o mesmo número de rótulo. Você deve programar essa instrução de entrada como a primeira instrução de uma linha. LBL sempre é avaliada como verdadeira ou 1 lógico. Os números de rótulos são únicos, isto é, não podem ser repetidos.

Saltar à frente para um rótulo reduz o tempo de varredura do programa ao omitir um segmento do programa até que seja necessário. Saltar para trás permite que o controlador execute repetidamente segmentos do programa.

Mais de uma instrução JMP pode saltar para o mesmo rótulo.

Obs. Tenha cuidado ao usar a instrução JMP para saltar para trás ou fazer loops em seu programa. Se você fizer loops muito demorados, o temporizador de controle pode exceder o limite de tempo e causar uma falha no processador. Use um contador, temporizador, ou registro de varredura do programa (S:3, bits 0-7) para limitar o tempo gasto dentro de loops com instruções JMP/LBL.

PARÂMETROS:

Digite um número decimal para o rótulo, de 0 a 999. Você pode colocar:

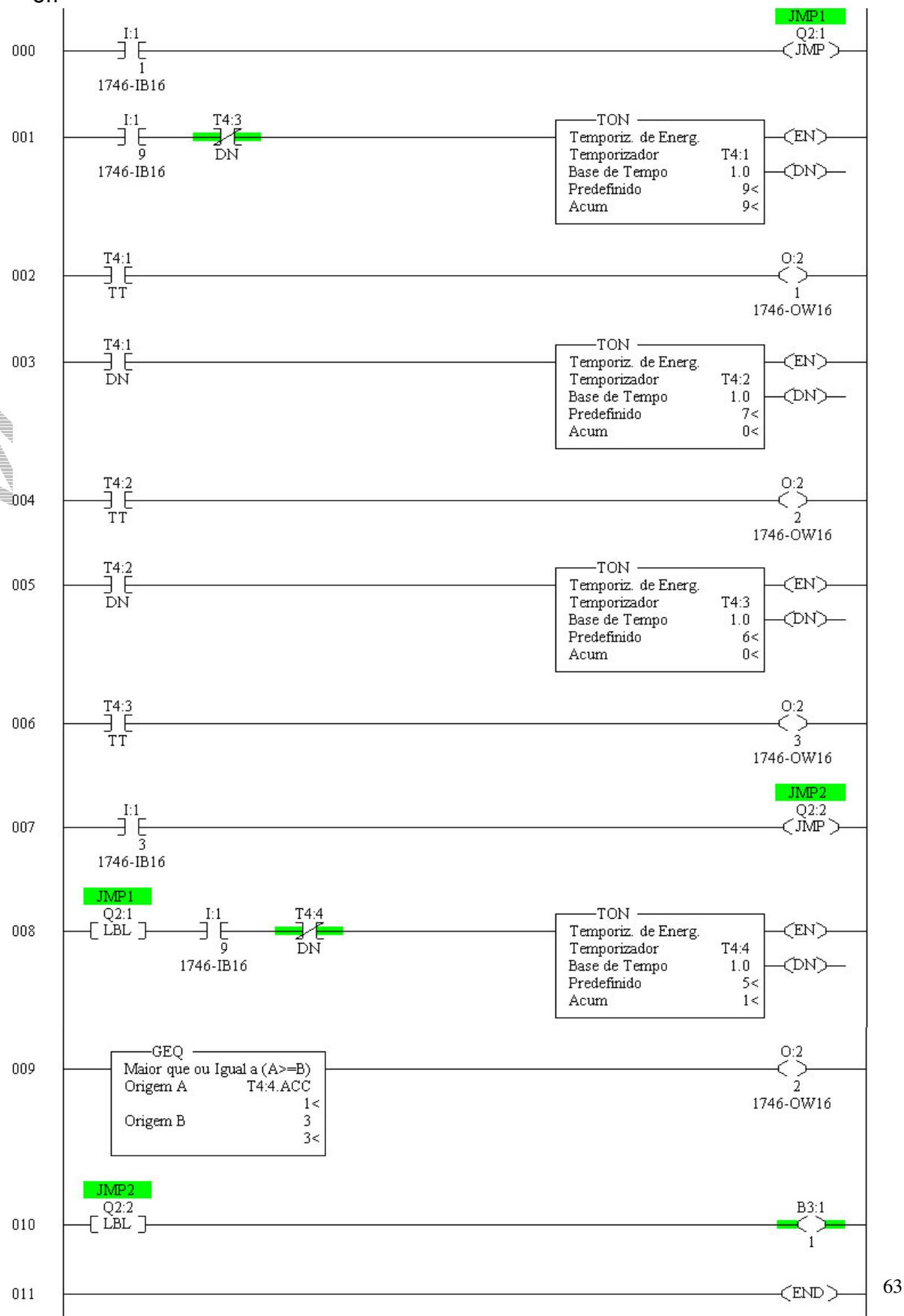
- Até 256 rótulos para controladores SLC em cada arquivo de subrotina.
- Até 1000 rótulos para controladores MicroLogix em cada arquivo de subrotina.

SENAC CLIMATEC

SENACINA

EXEMPLO:

1. Programa Semáforo e Pisca-pisca utilizando JMP e LABEL A seleção do funcionamento como semáforo ou como pisca-pisca é feita através de chaves on-off

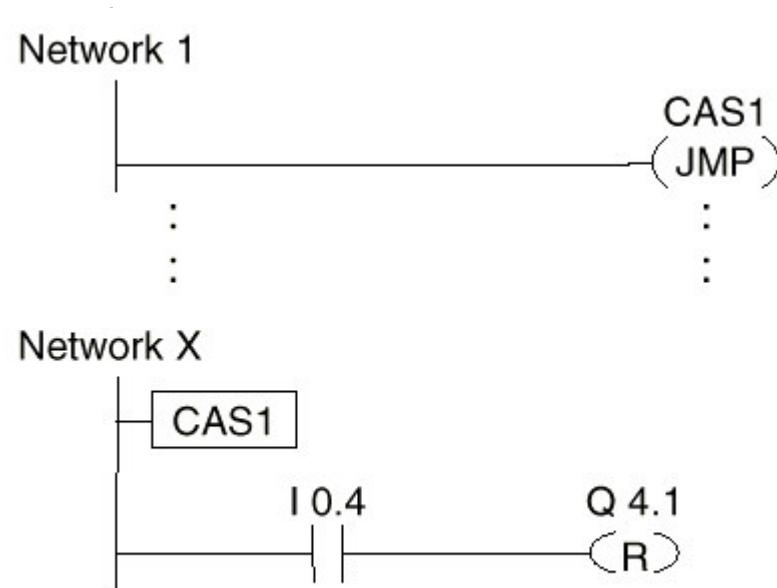


4.4.2 Para CLP SIEMENS

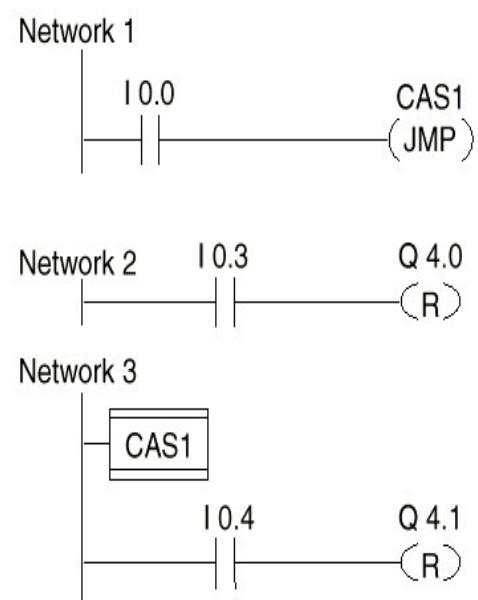
JUMP(Jump Within the Block When 1) – Pula para um determinado Rótulo(Label) quando o mesmo é executado. Quando inexiste condição para a sua execução esta instrução é chamada de incondicional. Quando ele executado após o programa atender uma determinada condição, esta instrução é chamada de Jump condicional.

Exemplo:

- 1- Jump Incondicional – O CLP vai para a posição CAS1 quando executa a Network 1. Ele não executará nenhuma Network entre o Jump e o rótulo.

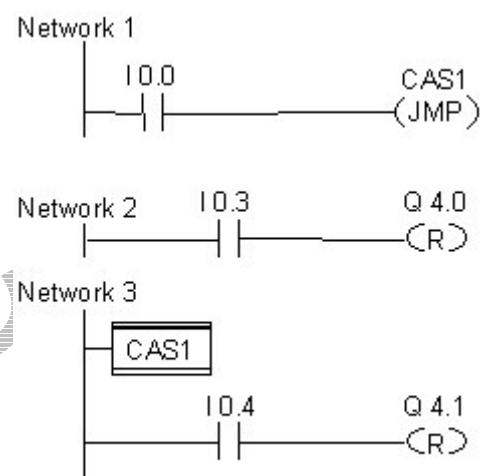


- 2- Jump Condicional – Quando a condição I0.0 for verdadeira o CLP irá pular todas as linhas entre o Jump e o rótulo(CAS1), executando as linhas subsequentes.



JMPN(Jump if Not)- Corresponde a negação do JMP. Esta instrução executa um pulo para o Rótulo se a condição de entrada for falsa.

Exemplo JMPN: Pula para o rótulo após quando a condição I0.0 for falsa



Label(Rótulo) – Identificador da posição que ocorrerá JMP. Para a colocação do nome do rótulo devemos iniciar sempre com um caractere alfabético.



Símbolo do Label

DESAFIO:

Crie um programa semáforo e pisca-pisca utilizando JMP. O semáforo deve funcionar das 6 às 24 h e o pisca-pisca das 0 às 6 h, automaticamente, a partir do relógio de tempo real do CLP.

4.5 Chamada de Subrotina

4.5.1 Para PLC Allen-Bradley

SBR [Subrotina]

Exemplo:



Uma subrotina serve para armazenar seções repetitivas da lógica do programa que devem ser executadas a partir de diversos pontos dentro de um projeto. Uma subrotina economiza memória porque você a programa apenas uma vez.

Colocada como a primeira instrução em um arquivo de subrotina, a instrução SBR identifica o arquivo. Esse é o número do arquivo usado na instrução JSR para identificar o alvo para onde o programa deve saltar.

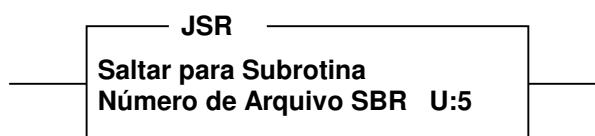
Essa instrução não tem bits de controle. Ela sempre é avaliada como verdadeira. A instrução deve ser programada como a primeira instrução da primeira linha de uma subrotina. O uso dessa instrução é opcional, porém é recomendado.

AVISO: As saídas controladas de dentro de uma subrotina permanecem no seu último estado até que a subrotina seja executada novamente.

Salto para uma Subrotina

JSR [Saltar para Subrotina]

Exemplo:



JSR é uma instrução de saída que faz com que o processador salte para o arquivo alvo da subrotina.

Você só pode saltar para a primeira instrução em uma subrotina. Cada subrotina deve ter um número de arquivo exclusivo (decimal, 3-255).

Aninhar subrotinas permite direcionar o fluxo do programa, do programa principal para uma subrotina e daí para outra subrotina. As seguintes regras aplicam-se quando aninhar subrotinas:

Processadores Fixo e 5/01 - você pode aninhar subrotinas até 4 níveis.
Processadores 5/02, 5/03, 5/04 e MicroLogix - você pode aninhar subrotinas em até 8 níveis.

RET [Retorno da Subrotina]

Exemplo:



Essa instrução de saída marca o final da execução da subrotina ou o final do arquivo de subrotina. Ela faz com que o processador retome a execução no arquivo do programa principal na instrução seguinte à instrução JSR onde ele saiu do programa. Se a seqüência de subrotinas aninhadas está envolvida, a instrução faz com que o processador retorne a execução do programa para a subrotina anterior.

Sem uma instrução RET, o comando END (sempre presente na subrotina) retorna automaticamente a execução do programa para a instrução JSR no seu programa de contatos que a chamou.

4.5.2 Chamada de Subirrotina para PLC SIEMENS

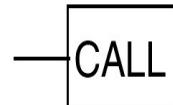
Para chamada de subirrotina O PLC da SIEMENS utiliza o Comando CALL_XX, onde XX está representando o programa chamado. Por exemplo:

CALL : Sem parâmetro, chama uma função.
CALL_FB : Chama um bloco de Funções(FB)
CALL_FC : Chama uma Função(FC)
CALL_SFB : Chama um sistema de FB
CALL_SFC: Chama um sistema de FC

Call Sem Parâmetros- Com a instrução **Call FC/SFC** sem parâmetros podemos chamar uma Função ou um Sistema de Funções que não vem com parâmetros. Na seção de codificação de uma função (FC), você na pode especificar qualquer parâmetro do tipo de Bloco_FC como endereço para uma chamada condicional. Você pode, contudo, um parâmetro do tipo Bloco_FC como o endereço de um bloco de funções (FB). Ao executar esta instrução ocorrerão as seguintes ações:

- O Endereço para retorno ao programa principal após a execução da subrotina será salvo.
- Os dados são salvos.
- Área para armazenar os dados locais da FC ou SFC.

<FC-/SFC number>



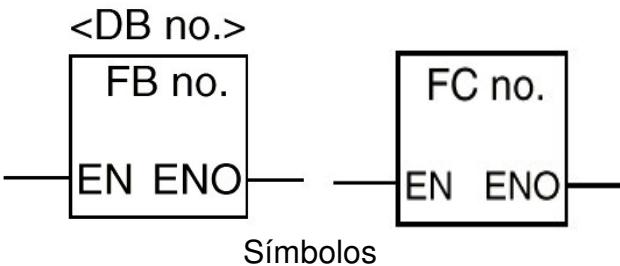
Símbolo da instrução sem Parâmetro.

CALL_FB (Call FB Como Box)

O símbolo depende da FB (se tem e quantos parâmetros existem). EN, ENO e o nome ou número da função.

Descrição

CALL_FB ou **CALL_FC** são executadas quando o estado do sinal EN é 1



Símbolos

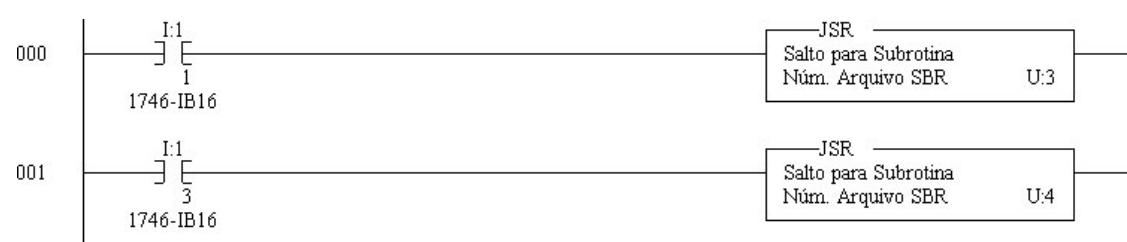
Onde Teremos os seguintes Parâmetros:

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M	Enable input
ENO	BOOL	I, Q, M	Enable output
FB no.	BLOCK_FB	-	Number of the FB/DB, range depends on the CPU
DB no.	BLOCK_DB	-	

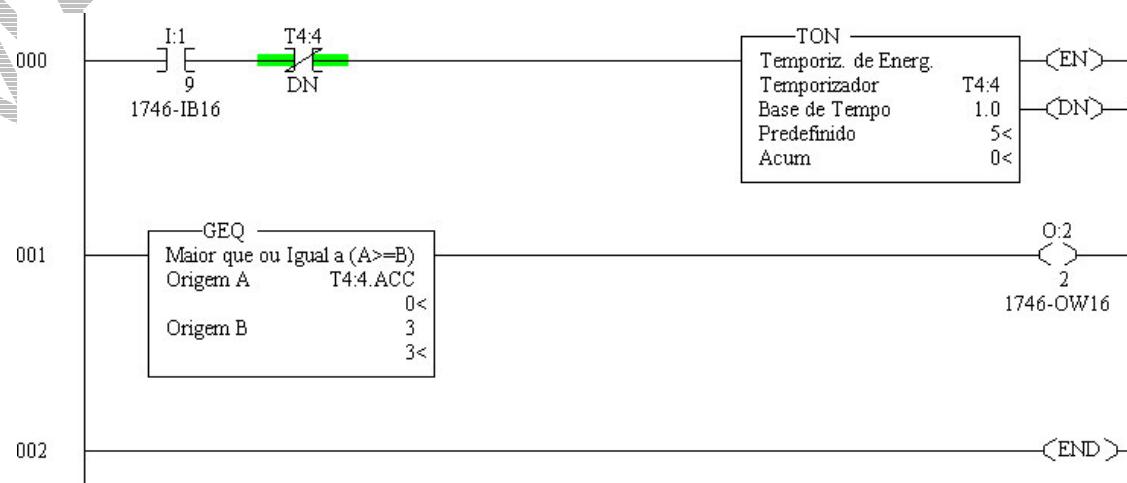
EXEMPLO:

Programa Semáforo e Pisca-pisca utilizando subrotinas JSR, SBR e RET. A seleção do funcionamento como semáforo ou como pisca-pisca é feita através de chaves on-off

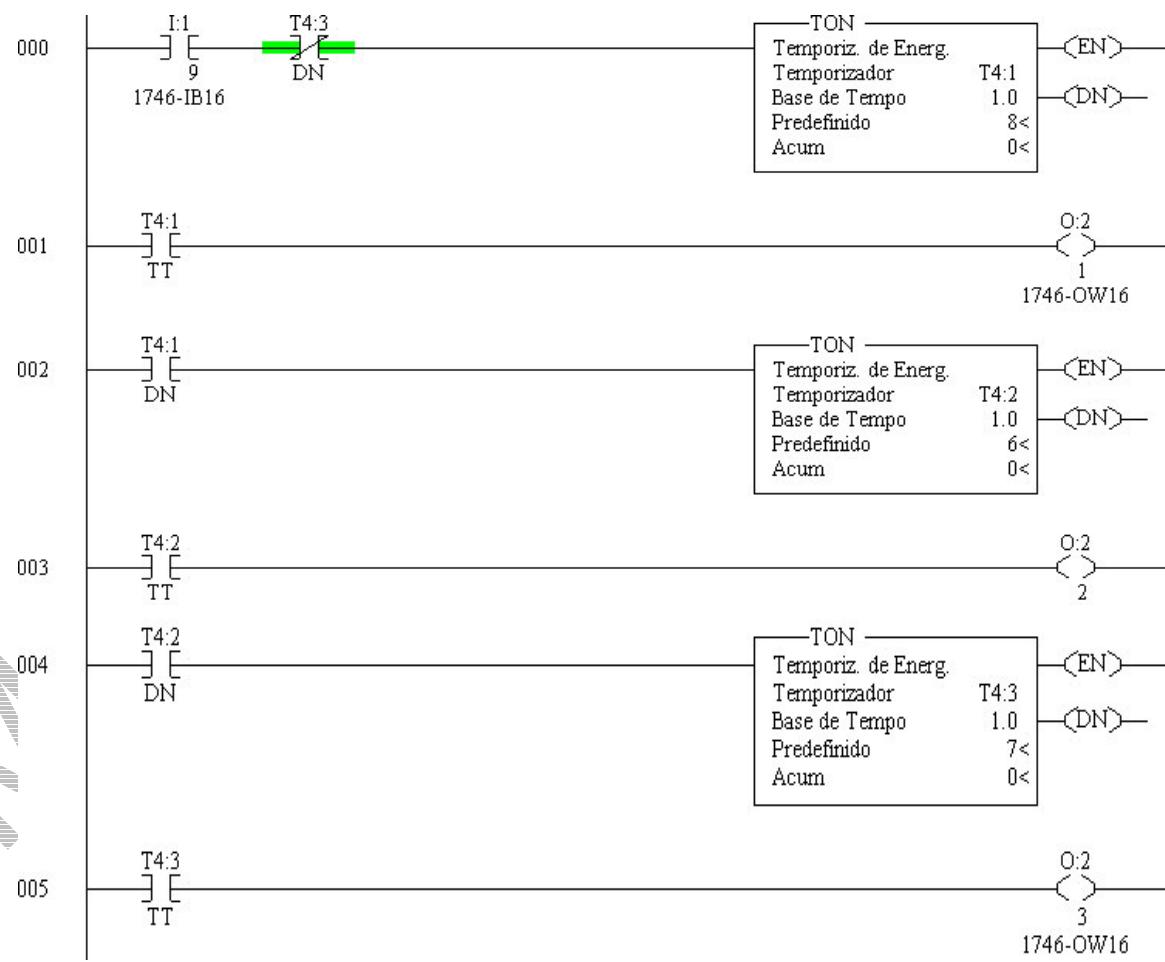
LAD 2 – Arquivo Principal



LAD 3 – PISCA-PISCA



LAD 4 – SEMÁFORO



DESAFIO:

Crie um programa Semáforo e Pisca-pisca utilizando subrotinas a instrução Call e RET. **O semáforo deve funcionar das 6 às 24 h e o pisca-pisca das 0 às 6 h, automaticamente, a partir do relógio de tempo real do CLP.**

4.6 SCL – Escala

4.6.1 Escala para CLP Allen-bradley

Exemplo:

SCL	
Escala	
Origem	I:9.1
Taxa [/10000]	4000
Deslocamento	100
Destino	N7:14

Essa instrução é utilizada para escalar dados de módulos analógicos e convertê-los para os limites prescritos pela variável de processo ou outro módulo analógico. Por exemplo, use SCL para converter um sinal de entrada de 4 a 20 mA para uma variável de processo PID. Ou use SCL para escalar uma entrada analógica para controlar uma saída analógica.

Quando as condições da linha são verdadeiras, essa instrução multiplica a origem por uma taxa especificada e depois divide por 10000. O resultado arredondado é adicionado a um valor de deslocamento e colocado no destino.

Equações usadas no cálculo do valor a ser colocado no destino:

$$\text{Destino} = (\text{Origem} \times (\text{taxa} / 10.000)) + \text{deslocamento}$$

Onde:

$$\text{Taxa} = 10.000 \times \frac{(\text{saída máxima} - \text{saída mínima})}{(\text{entrada máxima} - \text{entrada mínima})}$$

$$\text{Deslocamento} = \text{saída mínima} - \frac{\text{entrada mínima} \times \text{taxa}}{10.000}$$

Parâmetros:

Os valores devem estar entre -32768 e +32767 para os seguintes parâmetros.

Origem - deve ser um endereço de palavra.

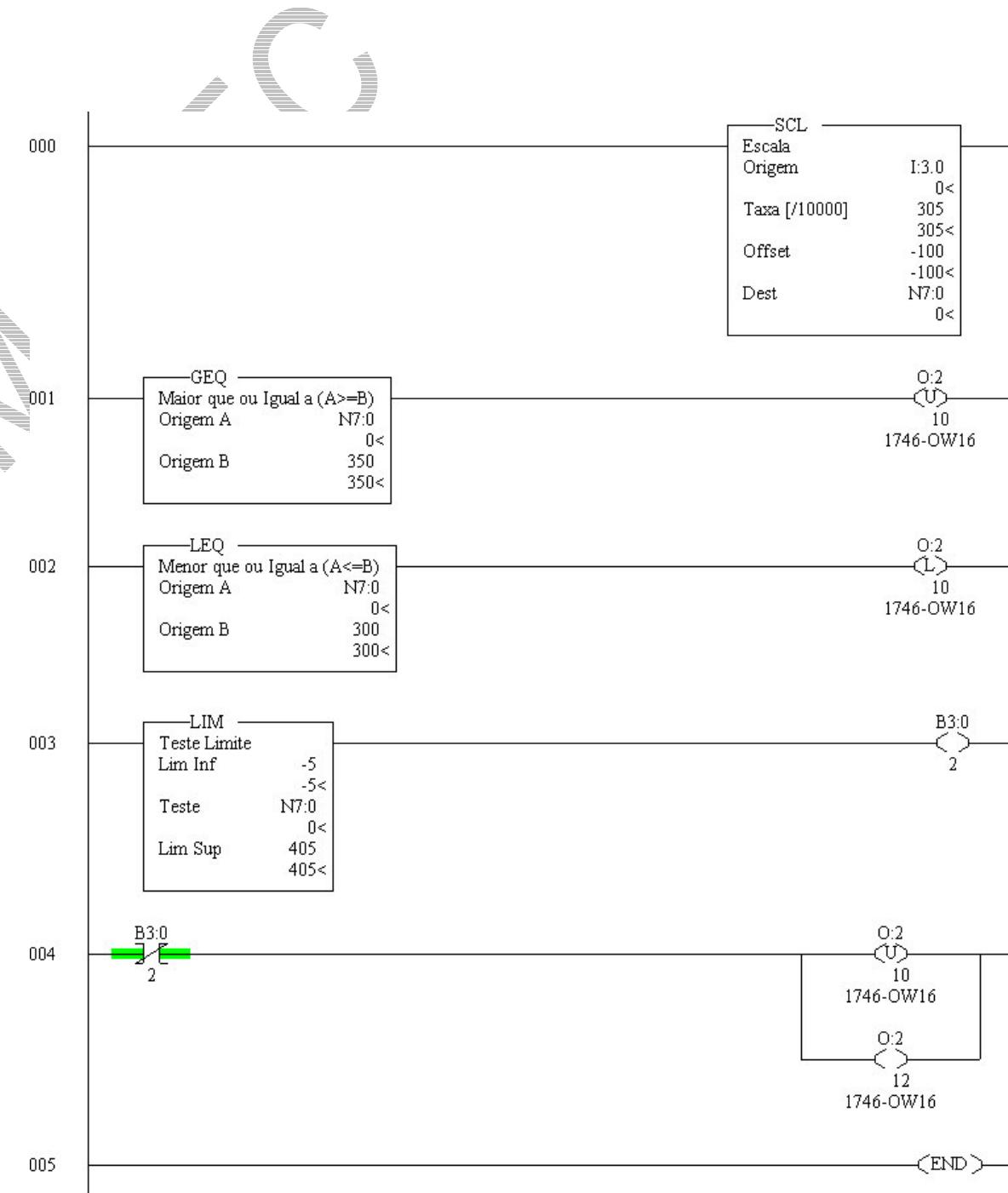
Taxa - (ou inclinação) é um valor positivo ou negativo. Pode ser uma constante de programa ou um endereço de palavra.

Deslocamento - pode ser uma constante de programa ou um endereço de palavra.

Destino - É o endereço do resultado da operação.

EXEMPLO:

Programa para realizar o controle on-off da temperatura de um forno. A medição de temperatura é realizada através de um transmissor cuja saída é no padrão 4 a 20 mA (zero vivo). Este transmissor é calibrado para a faixa de 0 a 400°C (0 a 100%). O aquecimento liga caso a temperatura caia até 300 °C e desliga se a temperatura subir até 350 °C. Caso haja algum problema no transmissor, o aquecimento é desativado e um alarme é acionado. O transmissor é simulado com o módulo de entrada e saída analógica. Este módulo fornece tensões de 0 a 10 V. Portanto, será utilizado o padrão de tensão da instrumentação de 1 a 5 V (0 a 100%)



4.6.2 Escala em CLP SIEMENS

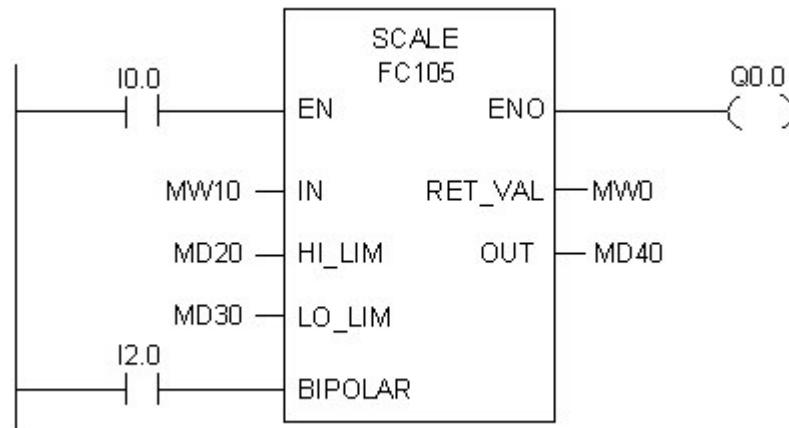
A função SCALE leva um número inteiro(IN) e converte para uma unidade de engenharia localizada entre um limite inferior(Lo) e um limite superior(Hi). O resultado desta operação é escrito na saída OUT e usa a seguinte equação:

$$OUT = \left[\frac{(FLOAT(IN) - K1)}{K2 - K1} * (HILIM - LOLIM) + LOLIM \right]$$

As constantes K1 e k2 são valores baseadas nas seguintes condições:

1 - Bipolar k1=-27648.0 e k2=+27648.0.

2 - UNIPOLAR: K1 = 0.0 and K2 = +27648.0



Símbolo do SCALE

4.6.3 UNSCALE para CLP SIEMENS

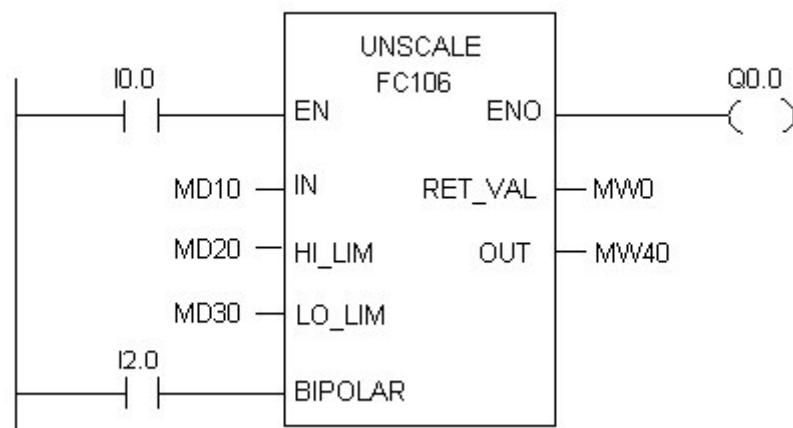
A função UNSCALE leva um número real de entrada (IN) em unidade de engenharia entre um limite inferior(LoLIM) e um limite superior(HiLIM) e converte ele para inteiro. Da seguinte forma.

$$OUT = \left[\left(\frac{(IN - LOLIM)}{(HILIM - LOLIM)} \right) * (K2 - K1) \right] + K1$$

Onde :

1 - BIPOLAR: A saída inteira estará entre -27648 e 27648, portanto, K1 = -27648.0 and K2 = +27648.0

2 - UNIPOLAR: A saída inteira estará entre 0 e 27648, contudo, K1 = 0.0 e K2 = +27648.0



Símbolo do UNSCALE

Crie um programa para realizar o controle on-off do **nível de um tanque de 1000 m³**. A medição de nível é realizada através de um transmissor cuja saída é no padrão 4 a 20 mA (zero vivo). Este transmissor é calibrado para a faixa de **0 a 1000 m³** (0 a 100%). **A bomba que enche o tanque liga caso o nível caia até 500 m³ e desliga se o nível subir até 900 m³**. Caso haja algum problema no transmissor, a bomba deve ser desativada e um alarme deve ser acionado. O transmissor é simulado com o módulo de entrada e saída analógica. Este módulo fornece tensões de 0 a 10 V. Portanto, será utilizado o padrão de tensão da instrumentação de 1 a 5 V (0 a 100%)

5 INSTRUÇÃO PID

Exemplo:

PID	
Bloco de Controle	N7:9
Variável de Processo	I:9.0
Variável de Controle	N7:11
Comprimento do Bloco de Controle	23
Tela de Configuração	

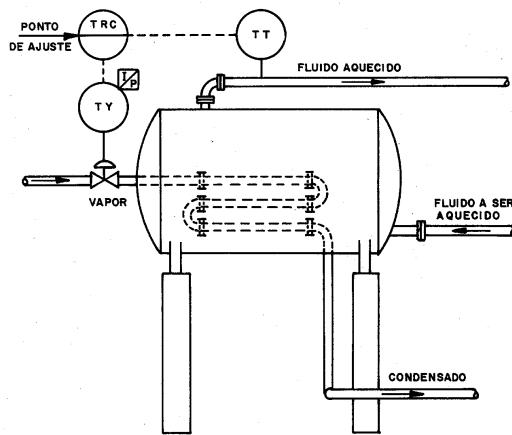
Essa instrução de saída é usada para controlar variáveis físicas como temperatura, pressão, nível de líquido ou vazão em malhas de controle de processo.

A instrução PID normalmente controla um malha fechada usando entradas de um módulo de entrada analógico e fornecendo uma saída para um módulo de saída analógico como uma resposta a uma variável de processo mantida efetivamente em determinado Set Point (ponto pré-programado).

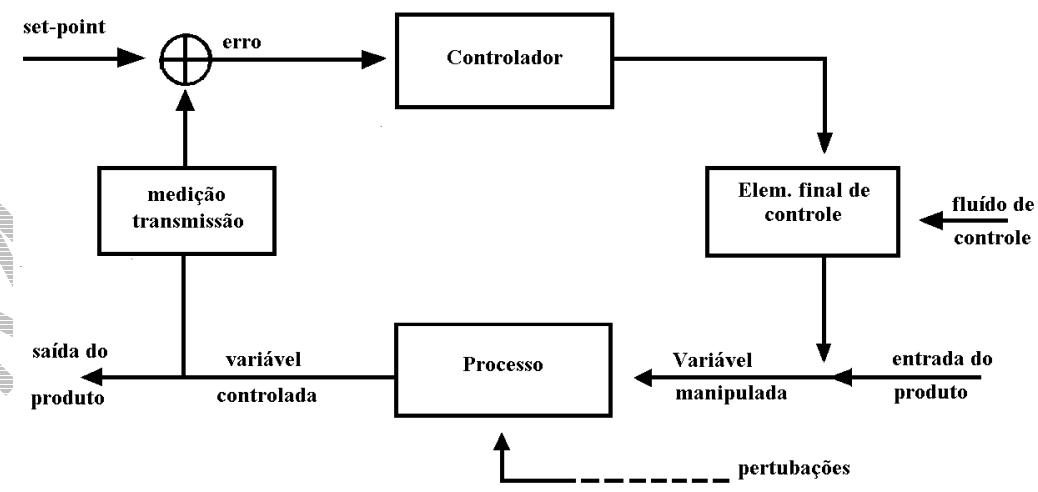
A equação PID controla o processo enviando um sinal de saída ao atuador. Quanto maior o erro entre o Set point e a entrada da PV (variável de processo), maior o sinal de saída e vice versa. Um valor adicional (feed forward ou polarização) pode ser adicionado à saída de controle como um patamar. O resultado do cálculo PID (variável de controle) irá dirigir a variável de processo que você está controlando, para o Set Point.

5.1 Conceito de PID

Esse é um exemplo de como opera uma malha PID simples. É um malha de controle de temperatura básico.



E o diagrama genérico da malha de controle



A equação PID controla o processo enviando um sinal de saída para uma válvula de controle. Quanto maior o erro entre o Set Point e a entrada da variável de processo, maior o sinal de saída e vice versa. Um valor adicional (feed-forward ou polarização) pode ser adicionado à saída de controle como um patamar. O resultado do cálculo PID (variável de controle) irá dirigir a variável de processo, que você está controlando, para o Set Point.

5.2 A Equação PID

A instrução PID usa o seguinte algoritmo:

$$CV = Kc \left[(E) + \frac{1}{Ti} \int E dt + Td \frac{dPV}{dt} \right]$$

As constantes de Ganho Padrão são:

Termo	Símbolo	Faixa (Inferior para Superior)
Ganho do Controlador	Kc	0,1 a 25,5 (sem dimensão) 0,01 a 327,67 (sem dimensão)*
Tempo Integral	Ti	25,5 a 0,1 (minutos por repetição) 327,7 a 0,01 (minutos por repetição)*
Tempo Derivativo	Td	0,01 a 2,55 (minutos) 0,01 a 327,67 (minutos)*

* Aplica-se a faixas PID do 5/03 e 5/04 quando o bit Redefinir Ganho (RG) é ativado.

O termo derivativo suaviza o sinal através de um filtro passa-baixas. A frequência de corte do filtro é 16 vezes maior que a frequência de corte do termo derivativo.

5.3 Parâmetros para o PID Allen-Bradley

Normalmente, você coloca a instrução PID em uma linha sem lógica condicional. A saída permanece no seu último valor quando a linha é falsa. O termo integral também é zerado quando a linha é falsa.

A instrução PID não permite valores de ponto flutuante para nenhum de seus parâmetros. Logo, se você tentar mover um valor de ponto flutuante para um dos parâmetros PID (com a instrução MOV, por exemplo), ocorre uma conversão de ponto flutuante para inteiro.

Bloco de Controle - um arquivo que armazena os dados necessários para operar a instrução. O comprimento do arquivo é fixo em 23 palavras e deve ser fornecido como um endereço de arquivo inteiro. Não grave em endereços de bloco de controle com outras instruções no seu programa. Apenas o set point os seguintes sinalizadores de instrução PID podem ser ativados ou zerados por seu programa de contatos:

SP (Set Point)	Palavra 2 do Bloco de Controle
TM (bit de modo temporizado)	Palavra 0 do Bloco de Controle, bit 0
AM (bit auto/manual)	Palavra 0 do Bloco de Controle, bit 1
CM (bit modo de controle)	Palavra 0 do Bloco de Controle, bit 2
OL (bit ativar limitação de saída)	Palavra 0 do Bloco de Controle, bit 3

AVISO!

Não altere o estado de nenhum valor de bloco de controle PID a menos que você entenda completamente sua função e efeitos relacionados em seu processo. Uma operação inesperada pode resultar em possíveis danos ao equipamento e/ou ferimentos pessoais.

Dica: Use um arquivo de dados exclusivo para seu bloco de controle PID (N9:0, por exemplo). Isso evita reutilização acidental dos endereços do bloco de controle PID por outras instruções no seu programa.

Comprimento do Bloco de Controle - Especifica um arquivo inteiro, por exemplo N7:0. O comprimento do arquivo é fixo de 23 palavras.

Variável de Processo PV - O endereço de elemento que armazena o valor de entrada do processo. Esse endereço pode ser o local da palavra de entrada analógica onde o valor do A/D de entrada é armazenado. Esse valor também pode ser um valor inteiro se você preferir pré-escalar seu valor de entrada para a faixa 0-16383.

Variável de Controle CV - O endereço de elemento que armazena a saída da instrução PID. A faixa do valor de saída vai de 0 a 16383, com 16383 sendo 100% do valor ON. Esse normalmente é um valor inteiro e você pode escalar a faixa de saída PID para a faixa analógica particular que seu aplicativo requerer.

Tela de Instalação - clique duas vezes no item *Tela de Instalação* para exibir uma tela que solicita a você outros parâmetros para completar a programação da instrução PID.

5.4 Tela de Instalação PID

Ao Clicar Tela de Configuração na instrução PID, aparece um diálogo que permite a entrada de parâmetros adicionais. Os parâmetros são descritos aqui.

Parâmetro	Faixa válida	Descrição
Kc Ganho do Controlador	0 até +327,67*	Este é o ganho proporcional da equação PID.
Ti Tempo Integral	0 até +327,67* minutos	Este é o tempo integral da equação PID.
Td Tempo Derivativo	0 até +327,67* minutos	Este é o tempo derivativo da equação PID.
Atualizar Circuito	0.01 a 10.24 segundos	Este é o intervalo de tempo entre cálculos PID. O valor é indicado em intervalos de 0,01 segundos. Geralmente, digite um tempo de atualização de circuito entre cinco e dez vezes mais rápido que o período natural de carga. No modo STI, este valor precisa equivaler ao valor do intervalo de tempo STI S:30.
Modo	Selecione E = SP - PV (Ação)	Ação Reversa causa um aumento no CV de saída quando o PV de entrada é menor que

Controle	Reversa) ou E = PV - SP (Ação Direta)	o set point SP (por exemplo, em uma aplicação de aquecimento). Ação direta causa um aumento no CV de saída quando o PV de entrada é maior que o set point SP (por exemplo, em uma aplicação de resfriamento).
Controle PID	Selecione Auto ou Manual	Auto indica que o PID controla a saída. Manual indica que o usuário define a saída.
Modo de Data/Hora	Selecione Temporizado ou STI	Com o modo Temporizado selecionado, o PID atualiza a sua saída a intervalos especificados no parâmetro de atualização de circuito. Ao usar o modo temporizado, o tempo de varredura do seu processador deve ser pelo menos dez vezes mais rápido do que o tempo de atualização de circuito para evitar imprecisões na temporização ou distúrbios Com o modo STI selecionado, o PID atualiza a sua saída a cada varredura da subrotina STI. Ao selecionar STI, a instrução PID deve ser programada em uma sub-rotina de STI de interrupção, e a rotina STI deve possuir um intervalo de tempo que equivale à definição do parâmetro de atualização do circuito PID. Defina o período STI na palavra S:3.0
Limitar Saída CV	Selecione Sim ou Não	Selecionar Sim limita a saída aos valores mínimo e máximo. Selecionar Não não aplica limites à saída.
Zona Morta DB	0 até o máximo escalado, ou entre 0 e 16383 quando não existe escala.	A zona morta se estende acima e abaixo do set point especificado por você. A zona morta só tem efeito depois que a variável de processo PV cruza o Set Point.

* Nota: O bit RG deve ser ativado para aceitar valores acima de 25,5 quando utilizar processadores 5/03 e 5/04.

Entradas

Parâmetro	Faixa válida	Descrição
Valor de Ref. SP	1 e 16383, ou dentro da faixa válida de unidades de engenharia	Set Point ou o ponto de controle desejado da variável do processo.
Val. Ref. MÁX (Vmáx)	-32768 até +32767	Se o Set Point carregar unidades de engenharia, isto corresponde ao valor do Set Point em unidades de engenharia quando a entrada de controle for 16383 (100%).

Val. Ref. MÍN (Vmín)	-32768 até +32767	Se o Set Point carregar unidades de engenharia, então este parâmetro corresponde ao valor do set point em unidades de engenharia quando a entrada de controle for 0 (0%).
Variável de Processo PV	(Não editável, só para visualização)	Este é o valor da variável de processo (a entrada analógica) em unidades de engenharia

Saída

Parâmetro	Faixa válida	Descrição
Controlar Saída CV (%)	0 a 100 %	Permite alterar a variável de controle de saída somente se você tiver selecionado o modo manual.
Saída Mín (CV%)	1 a 99 %	Se CV cair abaixo deste valor mínimo, o bit de alarme de limite inferior (LL) de saída será ativado. Se Limitar Saída CV for Sim, o valor que você digitar será a porcentagem mínima de saída que a variável de controle CV atingirá.
Saída Máx. (CV%)	1 a 99 %	Se CV exceder este valor máximo, o bit de alarme de limite superior (UL) de saída será ativado. Se Limitar Saída CV for Sim, o valor que você digitar será a porcentagem máxima de saída que a variável de controle CV atingirá.
Erro de Escala SE	(Não editável, só para visualização)	Este é o erro da equação PID (E = SP - PV ou E = PV - SP).

5.5 Bloco de Controle PID

Formato do Bloco de Controle:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Palavra 0	E N		D N	P V	S P	LL	UL	D B	D A	TF S C	S C G	R O L	O C M	A M	TM	
Palavra 1	Sub Código de Erro PID (MSB)															
Palavra 2	Ponto Pré-programado SP – Set Point															
Palavra 3	Ganho Kc															
Palavra 4	Tempo Integral Ti															
Palavra 5	Tempo Derivativo Td															
Palavra 6	Polarização à Frente (bias)															

Palavra 7	Set Point Máximo (Smax)
Palavra 8	Set Point Mínimo (Smin)
Palavra 9	Banda morta
Palavra 10	APENAS PARA USO INTERNO - NÃO ALTERE!!
Palavra 11	Saída Máxima
Palavra 12	Saída Mínima
Palavra 13	Atualizar Circuito
Palavra 14	Variável de Processo Escalada
Palavra 15	Erro de Escala SE
Palavra 16	CV% de Saída (0-100%)
Palavra 17	Soma Integral MSW
Palavra 18	Soma Integral LSW
Palavra 19	APENAS PARA USO INTERNO - NÃO ALTERE!!
Palavra 20	APENAS PARA USO INTERNO - NÃO ALTERE!!
Palavra 21	APENAS PARA USO INTERNO - NÃO ALTERE!!
Palavra 22	APENAS PARA USO INTERNO - NÃO ALTERE!!

AVISO!

Não altere o estado de nenhum valor de bloco de controle PID a menos que você entenda completamente sua função e efeitos relacionados em seu processo. Uma operação inesperada pode resultar em possíveis danos ao equipamento e/ou ferimentos pessoais.

5.6 Indicadores de Status

Os seguintes indicadores de status associados com a instrução PID aparecem como **Marcadores** do lado direito da tela de **Configuração PID**. Acesse esta tela ao clicar **Tela de Instalação** na instrução PID.

Bits que podem ser ativados ou desativados por instruções no programa de contatos.

TM Bit de Modo de Data/Hora (palavra 0, bit 0)	Especifica o modo PID. É 1 para o modo TEMPORIZADO e 0 para o modo STI.
AM Bit Auto/Manual (palavra 0, bit 1)	Especifica a operação automática quando 0 e a operação manual quando 1.
CM Bit de Modo de Controle (palavra 0, bit 2)	Este bit é 0 se o controle for E=SP-PV. É 1 se o controle for E=PV-SP.
OL Bit de Limitar Saída Ativado (palavra 0, bit 3)	Este bit deve ser 1 se você optar por limitar a variável de controle.

Bits configurados pelo usuário

RG Bit Redefinir Ganho (palavra 0, bit 4)	Quando 1, este bit faz com que o valor de Ti e Kc sejam aumentados por um fator de 10 (o multiplicador de Kc e Ti muda para 0,01).
--	--

	Quando 0, este bit permite que T_i e K_c usem as mesmas faixas do PID do 5/02 (multiplicador de T_i e K_c de 0,1). Note que o Multiplicador de T_d não é afetado por esta seleção
DA Bit de Ação Derivativa (palavra 0, bit 7)	Quando 1, os cálculos do termo derivativo da equação do PID são feitos sobre o erro. Quando 0, os cálculos usam a PV.

Bits que apenas indicam condições da instrução PID

SC Sinalizador de Escala de Set Point (palavra 0, bit 5)	Quando 1 indica que os valores mínimo e máximo do Set Point não foram especificados
TF Tempo de Atualização de Circuito Muito Rápido (palavra 0, bit 6)	Este bit é ativado pelo algoritmo PID se o tempo de atualização de circuito especificado não puder ser atingido pelo programa (devido a limitações de tempo de varredura).
DB Erro de Zona Morta (palavra 0, bit 8)	Ativado quando a variável de processo está dentro da faixa da zona morta ao cruzar com 0
UL Alarme de Saída, Limite Superior (palavra 0, bit 9)	Ativado quando CV de saída de controle calculado excede o limite superior de CV.
LL Alarme de Saída, Limite Inferior (palavra 0, bit 10)	Ativado quando CV de saída de controle calculado é menor que o limite inferior de CV.
SP Set point Fora da Faixa (palavra 0, bit 11)	Ativado quando o set point excede o valor máximo escalado ou é menor que o valor mínimo escalado.
PV Variável de Processo Fora da Faixa (palavra 0, bit 12)	Ativado quando a variável de processo não escalada (ou bruta) excede 16383 ou é menor que zero.
DN PID Concluído (palavra 0, bit 13)	Este bit é ativado em varreduras onde é computado o algoritmo PID. É computado à taxa de atualização do circuito.
EN PID Ativado (palavra 0, bit 15)	Este bit é ativado enquanto a linha da instrução PID estiver ativada

5.7 Bloco PID para PLCs SIEMENS.

O SFB41/FB41 “Cont_C” é usado no CLP SIMATIC S7 para controlar processo com variáveis de entrada e saídas contínuas. Este controlador pode ser parametrizado facilmente através da ferramenta disponível no pacote Simatic Step7 chamado de Assign PID Control Parameter.

Descrição

Considerando que as funções no setpoint e desvio de processo, este PID implementa um controlador PID completo com variáveis manipuladas de saída e as opções de manipulações das variáveis manualmente.

Abaixo faremos uma breve descrição de algumas sub-funções:

Desvio do SetPoint – o setpoint é uma variável de entrada formatada como ponto flutuante na entrada SP_INT.

Desvio da variável de Processo - A variável do processo pode ser entrada no periférico no formato de ponto flutuante. O bloco CRP_IN é um bloco que converte esta variável para o formato de -100% a +100%.

a PV_NORM normaliza a saída de CRP_IN.

Sinal de Erro é a diferença entre o setpoint e a variável do processo. Para suprimir pequenos distúrbios é aplicado a um supressor de ruído chamado de DEADBAND. Se o parâmetro DEADB_W=0, o DEADBAND é desligado.

Algoritmo PID - A ação integral (INT) e derivativa (DIF) são conectadas em paralelo e pode ser ativada e podem ser ativadas e desativadas individualmente. Isto permite configurarmos este elemento com as ações P, PI, PD e PID. As condições Integral ou derivativa pura são também possíveis.

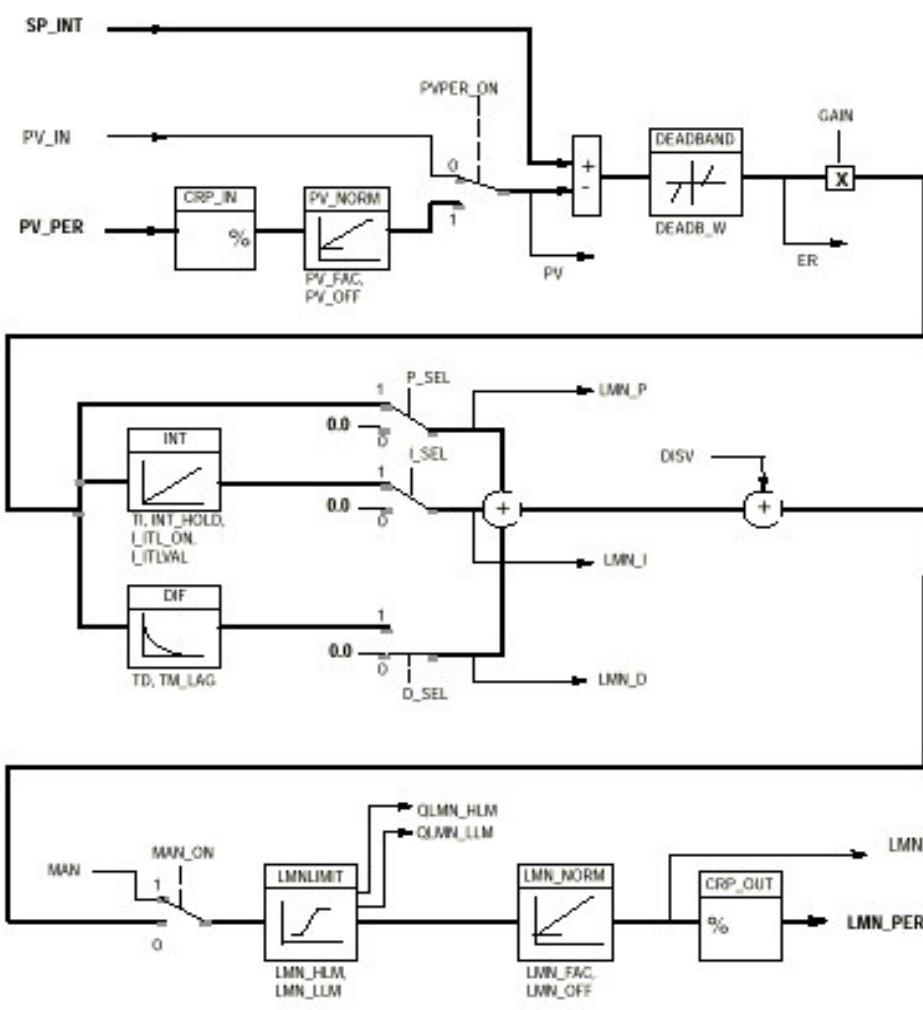
É também possível configurar este elemento para trabalhar no modo manual. Neste modo a variável manipulada é corrigida manualmente. A parte integral é ajustada internamente LMN_I-LMN_P-DISV e a parte derivativa para 0. isto significa que não haverá alteração brusca na saída do controlador quando passarmos para o modo automático.

Valores Manipulados podem ser limitados usando a função LMNLIMIT. Bits de sinalização indicam quando o limite é excedido. LMN_FAC default é 1 e LMN_OFF default é 0.

O valor de manipulado é também disponível no formato do periférico. O CPR_OUT converte o valor limitado para um valor do periférico.

Inicialização – Este bloco PID possui uma rotina de inicialização que roda quando o parâmetro COM_RST está em 1. na inicialização, o integrador é ajustado para o valor inicial I_ITVAL. Todas as outras saídas são ajustadas para os seus valores default. Abaixo apresentamos o diagrama em bloco deste PID,

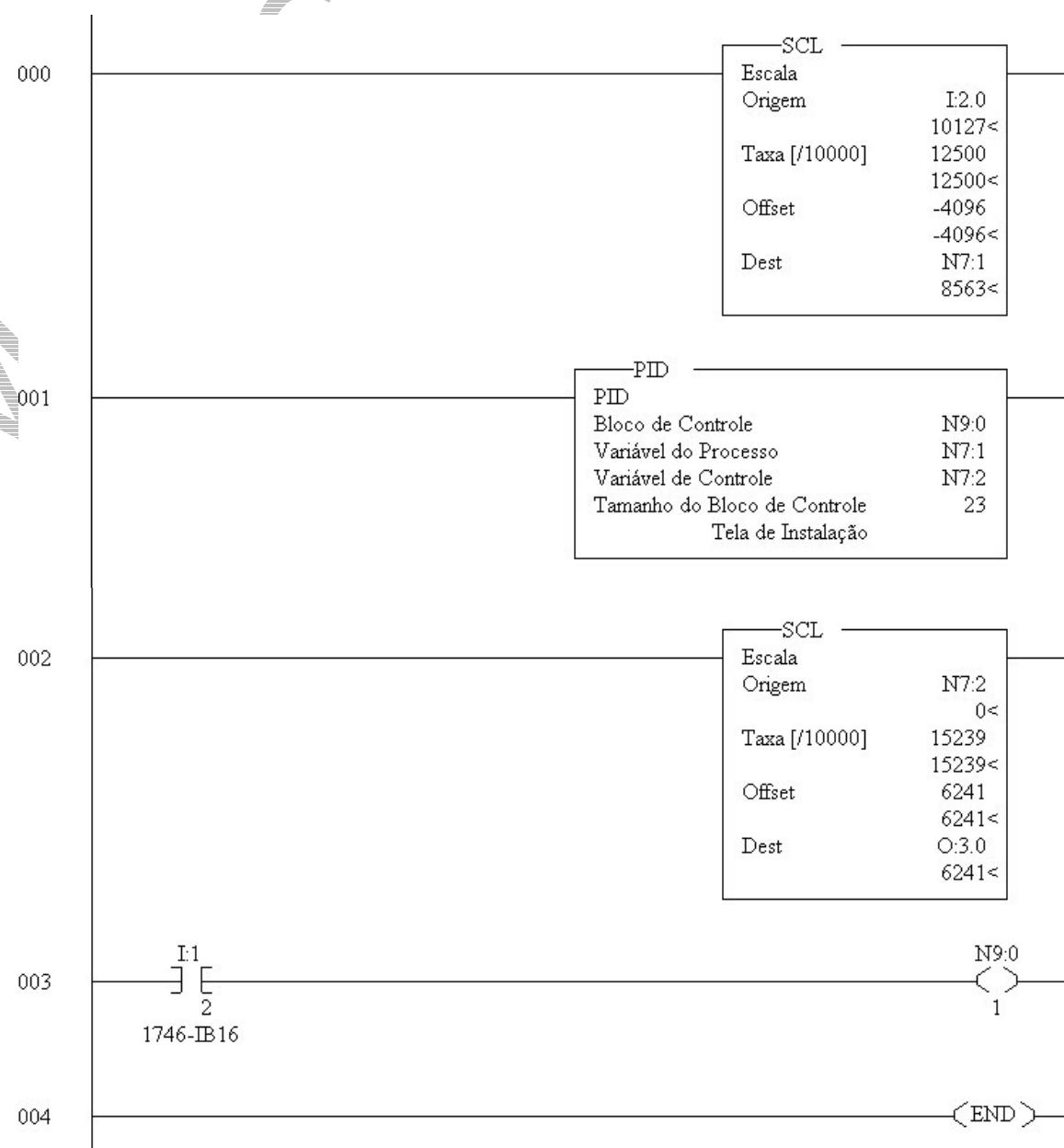
SENACIMA



EXEMPLO:

Programa para realizar o controle PID da temperatura de uma caldeira. A medição de temperatura é realizada através de um transmissor cuja saída é no padrão 4 a 20 mA (zero vivo). Este transmissor é calibrado para a faixa de 0 a 100°C (0 a 100%). O operador pode selecionar o modo de operação do controle entre automático ou manual através de uma chave on-off. O transmissor e o elemento final de controle são simulados com o módulo de entrada e saída analógica. Este módulo fornece tensões de 0 a 10 V(entrada analógica) e mede tensões de 0 a 10 V sobre um resistor de 250 Ω (saída analógica). Portanto, será utilizado o padrão de tensão da instrumentação de 1 a 5 V (0 a 100%).

SENAC/IMA



DESAFIO:

No programa exemplo da instrução PID, acrescente as instruções para, caso haja algum problema no transmissor, desligar o aquecimento e soar um alarme. Acrescente também as instruções para o operador possa alterar o Set Point através de uma chave codificadora (Thumbwheel), simulada através de chaves on-off.

6 Movimentação de Dados

As instruções de movimentação de dados são instruções de saída, ou seja, só são executadas quando as condições da linha são verdadeiras.

6.1 Copiar Arquivo

6.1.1 Copiar arquivo CLP Allen-Bradley

Exemplo

COP	
Copiar Arquivo	
Origem	#B3:80
Destino	#B3:20
Comprimento	23

Quando as condições da linha são verdadeiras para essa instrução de saída, um arquivo de origem definido pelo usuário é copiado para um arquivo destino.

Os elementos de origem e destino podem ser de um tipo diferente, mas o tipo de arquivo de destino determina quantas palavras de dados serão transferidas.

Parâmetros

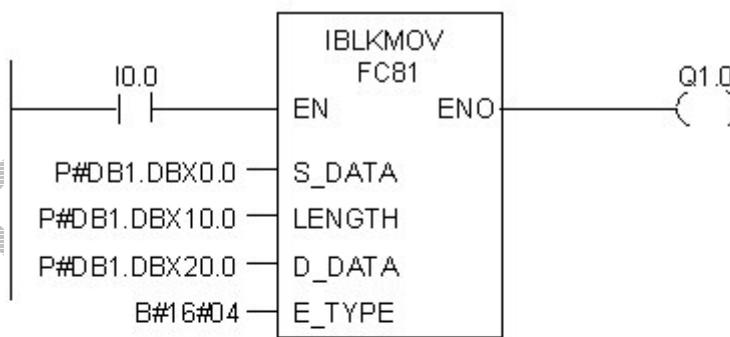
Origem - o endereço do arquivo você deseja copiar. Use o indicador de arquivo (#) no endereço.

Destino - o endereço inicial onde o arquivo de origem é copiado. Use o indicador de arquivo (#) no endereço.

Comprimento - o número de elementos no arquivo que você deseja copiar. Pode ser especificado um comprimento máximo de 128 palavras.

6.1.2 Copia arquivo Pelo CLP SIEMENS(FC81).

A função IBLKMOV é usada para mover um bloco de dados consistindo de bytes, palavra(W), inteiros(I), palavras duplas(DW) ou Duplo inteiro(DI) de um bloco origem para um bloco destino. O número de elementos para ser movido é determinado pelo comprimento (LENGTH) e o tamanho do arquivo é determinado pelo tipo de dado. O (S_DATA) e o (D_DATA) apontam para o local de inicio do movimento da origem e destino dos dados.



Before execution:

S_DATA → DBX0.0 = P#DB1.DBX50.0
DBW50 = W#16#2424
DBW52 = W#16#2525

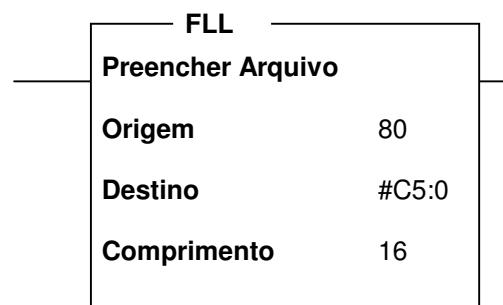
LENGTH → DBW10 = W#16#0002

D_DATA → DBX20.0 = P#DB2.DBX10.0
DBW10 = W#16#0000
DBW12 = W#16#0000

6.2 Preencher Arquivo

6.2.1 Para CLP Allen-Bradley(FLL)

Exemplo:



Essa instrução de saída preenche as palavras de um arquivo com um valor da origem.

Os elementos são preenchidos em ordem crescente até alcançar o número de elementos (o comprimento) ou até alcançar o último elemento do arquivo de destino, o que ocorrer primeiro.

O tipo de arquivo de destino determina o número de palavras por elemento que a instrução transfere. Por exemplo, se o tipo de arquivo de destino for contador e o tipo de arquivo de origem for inteiro, três palavras inteiras são transferidas para cada elemento no arquivo tipo contador.

Nenhuma conversão de dados ocorre se os arquivos de origem e de destino são de tipos diferentes; use o mesmo tipo de arquivo para os dois.

Parâmetros:

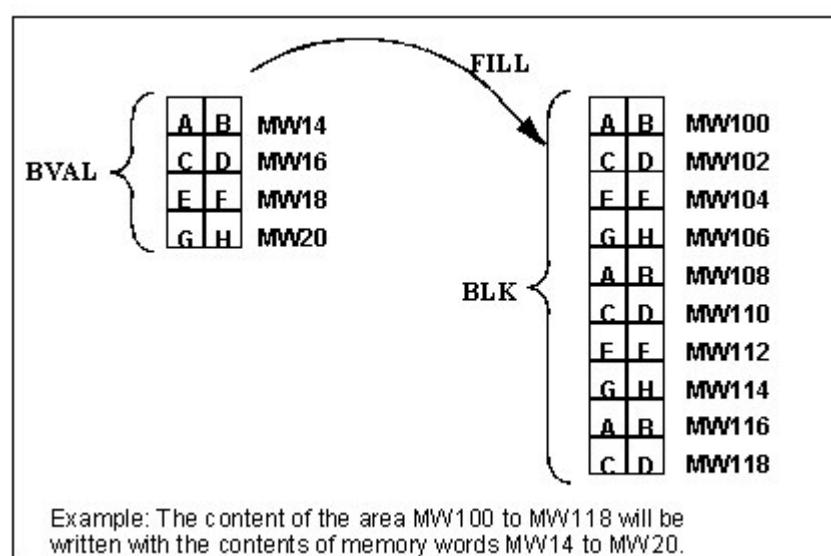
Origem - a constante de programa ou endereço do elemento. O indicador de arquivo (#) não é necessário para um endereço de elemento.

Destino - o endereço do arquivo de destino. A instrução grava sobre quaisquer dados já armazenados no destino.

Comprimento - o número de elementos no arquivo que você quer preencher. você pode especificar um comprimento máximo de 128 palavras.

6.2.2 Preenchimento para CLP SIEMENS(SFC21)

Com esta instrução você pega um determinado conteúdo e colo em outro até que a memória destino esteja completamente Cheia.

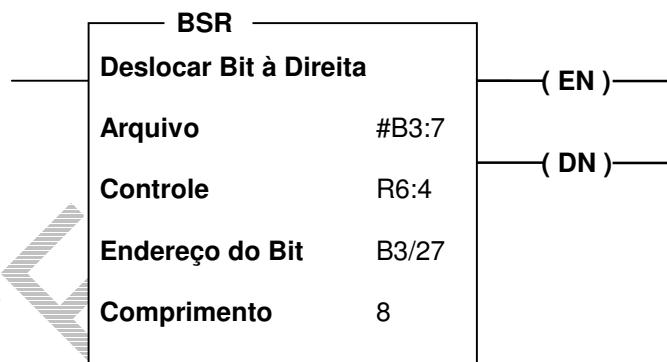


Exemplo de um comando FILL

6.3 Deslocar Bit à Direita e Deslocar Bit à Esquerda

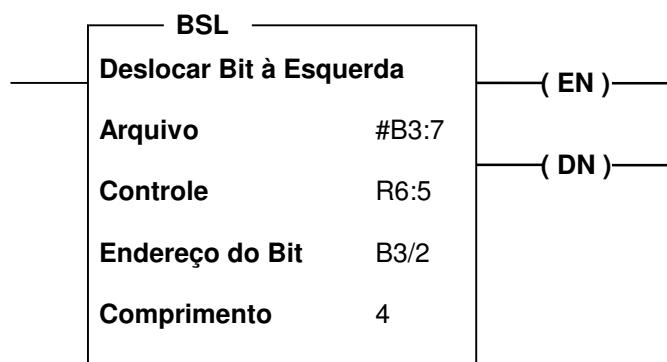
6.3.1 Deslocar Bit para o CLP Allen-bradley

Exemplo:



Em cada transição desligado-para-ligado na entrada, essa instrução de saída carrega um bit de dados em um bloco de bits, desloca o bloco para a direita e descarta o bit final .

Exemplo:



Em cada transição desligado-para-ligado na entrada, essa instrução de saída carrega um bit de dados em um bloco de bits, desloca o bloco para a esquerda e descarta o bit final .

Um exemplo do uso dessas instruções é acompanhar garrafas em uma linha de engarrafamento onde cada bit representa uma garrafa.

Parâmetros

Arquivo - esse é o endereço do bloco de bits que você deseja deslocar. Você deve usar o indicador de arquivo (#) no endereço do bloco de bits. . Você deve iniciar o bloco no limite do elemento de 16 bits, por exemplo, use o bit 0 do elemento Núm1, 2, 3 etc.

Controle - Esse é o endereço exclusivo da estrutura de controle (48 bits, 3 palavras de 16 bits) na área de controle da memória que armazena os bits de status da instrução, o tamanho do bloco (em número de bits), e o apontador do bit (atualmente não em uso).

O elemento de controle:

Palavra 0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	E N	D N		E R	U L											Não Usado
Tamanho do bloco de bits (número de bits)																
Apontador do Bit (atualmente não em uso)																

Bits de Status da Palavra de Controle

UL (descarregar) – bit de saída do bloco após o deslocamento.

ER (erro) – indica que um erro ocorreu, como um número negativo para o comprimento ou posição

DN (pronto) – indica que um deslocamento foi realizado.

EN (ativar) – ativado quando a linha de entrada passa de falso para verdadeiro.

Endereço do Bit - o local do bit que será adicionado ao bloco.

Comprimento - o número total de bits a ser deslocado. Podem ser até 2048 para o SLC e até 1680 para o MicroLogix 1000

6.3.2 Desloca Bits para o CLP SIEMENS

Você pode usar a instrução Shift para mover o conteúdo da entrada IN bit a bit para direita ou esquerda. Deslocar n bits para a esquerda multiplica o conteúdo de IN por 2 elevado a n

$$(IN * 2^n)$$

deslocar n bits para a direita significa dividir o conteúdo de IN por 2 elevado a n

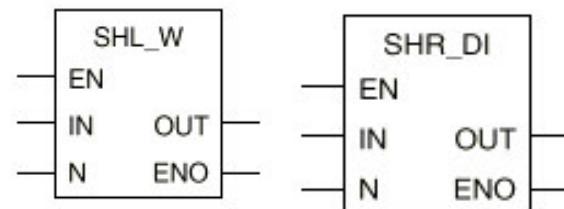
$$\frac{IN}{2^n}$$

Por exemplo, se você desloca o equivalente a 3 em binário, para esquerda de 3 bits, você obtém o equivalente a 24 em decimal. Se você desloca o equivalente a 16

em binário, 2 bits para direita , você obtém o equivalente a 4 em decimal. O número que você coloca a entrada N indica o número de bits a ser deslocado. A posição que ficará vaga após o deslocamento é preenchido com o bit que representará o sinal do número(0 para positivo ou 1 para negativo). O valor do último bit deslocado é armazenado dentro do bit CC1 na palavra de status. O bit CCO e o OV da palavra de status levado para 0. você pode usar a instrução de JUMP para avaliar o bit CC1.

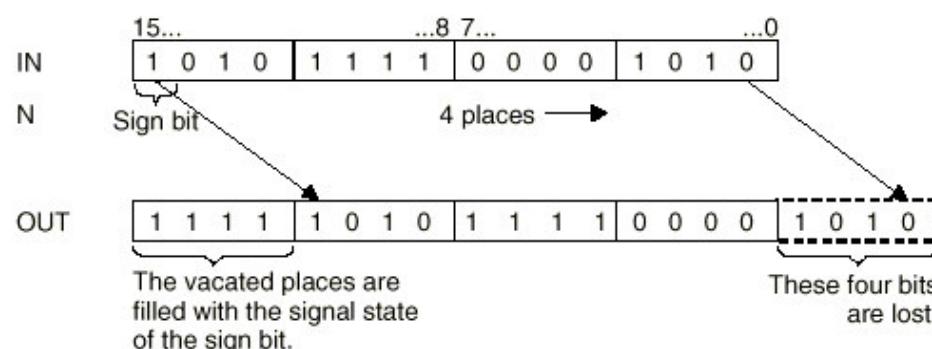
As seguintes instruções de SHIFT estão disponíveis:

- SHR_I : Desloca um número inteiro para direita.
- SHR_DI : Desloca um inteiro duplo para direita.
- SHL_W : Desloca para esquerda uma palavra.
- SHR_W : Desloca para direita uma palavra.
- SHL_DW : Desloca para esquerda uma palavra dupla.
- SHR_DW : Desloca uma palavra Dupla para direita.



Símbolos para a instrução Shift

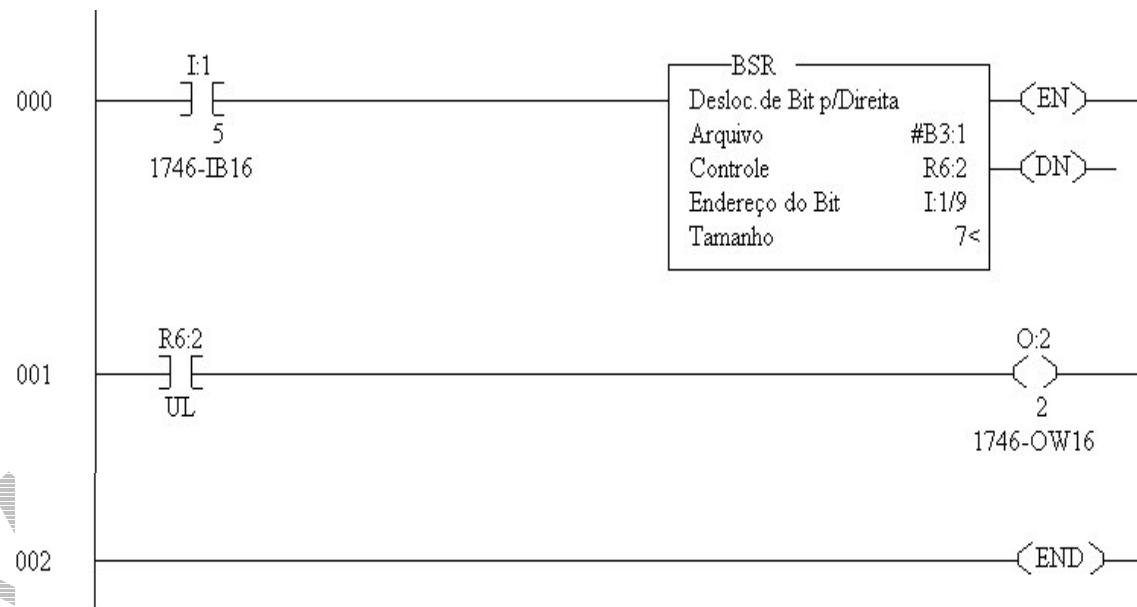
Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D, T, C	Enable input
IN	INT	I, Q, M, L, D	Value to be shifted
N	WORD	I, Q, M, L, D	Number of bit positions by which the value will be shifted
OUT	INT	I, Q, M, L, D	Result of the shift instruction
ENO	BOOL	I, Q, M, L, D	Enable output



Exemplo de um Deslocamento para Direita

EXEMPLO:

Programa que detecta e descarta garrafas com a boca ou o fundo quebrados em uma linha de engarrafamento, utilizando as instruções BSR ou BSL. Os sensores de garrafa (I:1/5) e de boca de garrafa (I:1/9) são simulados com chaves on-off e o atuador de descarte é simulado com uma lâmpada.



DESAFIO:

Acrescente, no programa exemplo de movimentação de dados, as instruções que realizem a contagem do número de garrafas quebradas, acionando um alarme quando este número atingir 20 unidades.

SENAC CLIMATEC

SENNA CLIMATEC

ANEXO 1

SISTEMAS DE NUMERAÇÃO.

Todos nós, quando ouvimos pronunciar a palavra números, automaticamente a associamos ao sistema decimal com o qual estamos acostumados a operar. Este sistema está fundamentado em certas regras que são bases para qualquer outro.

Vamos, portanto, estudar estas regras e aplicá-las aos sistemas de numeração *binária*, *octal* e *hexadecimal*. Estes sistemas são utilizados em computadores digitais, circuitos lógicos em geral e no processamento de informações dos mais variados tipos. O número decimal **573** pode ser também representado da seguinte forma:

$$573_{10} = 500 + 70 + 3 \text{ ou } 573_{10} = 5 \times 10^2 + 7 \times 10^1 + 3 \times 10^0$$

Isto nos mostra que um dígito no sistema decimal tem, na realidade, dois significados. Um, é o valor propriamente dito do dígito, e o outro é o que esta relacionado com a posição do dígito no número (peso). Por exemplo: o dígito **7** no número acima representa 7×10 , ou seja 70, devido a posição que ele ocupa no número. Este princípio é aplicável a qualquer sistema de numeração onde os dígitos possuem "pesos", determinados pelo seu posicionamento. Sendo assim, um sistema de numeração genérico pode ser expresso da seguinte maneira:

$$N = d_n \cdot B^n + \dots + d_3 \cdot B^3 + d_2 \cdot B^2 + d_1 \cdot B^1 + d_0 \cdot B^0$$

Onde:

N = Representação do número na base B

d_n = Dígito na posição n

B = Base do sistema utilizado

n = Valor posicional do dígito

por exemplo, o número **1587** no sistema decimal é representado como:

$$N = d_3 \cdot B^3 + d_2 \cdot B^2 + d_1 \cdot B^1 + d_0 \cdot B^0$$

$$1587_{10} = 1 \cdot 10^3 + 5 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0$$

Sistema de Numeração Binário

O sistema binário utiliza dois dígitos (**base 2**), para representar qualquer quantidade. De acordo com a definição de um sistema de numeração qualquer, o número binário **1101** pode ser representado da seguinte forma:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$1101_2 = 8 + 4 + 0 + 1 = 13_{10}$$

Note que os índices foram especificados em notação decimal, o que possibilita a conversão binária-decimal como descrito acima.

Através do exemplo anterior, podemos notar que a quantidade de dígitos necessários para representar um número qualquer, no sistema binário, é muito maior quando comparada ao sistema decimal. A grande vantagem do sistema binário reside no fato de que, possuindo apenas dois dígitos, estes são facilmente representados por uma chave aberta e uma chave fechada ou, um relé ativado e um relé desativado, ou, um transistor saturado e um transistor cortado; o que torna simples a implementação de sistemas digitais mecânicos, eletromecânicos ou eletrônicos.

Em sistemas eletrônicos, o dígito binário (**0 ou 1**) é chamado de *BIT*, enquanto que um conjunto de 8 bits é denominado *BYTE*.

• Conversão Binário - Decimal

A conversão de um número do sistema binário para o sistema decimal é efetuada simplesmente adicionando os pesos dos dígitos binários 1, como mostra o exemplo a seguir

a) 11010_2

b) 1100100_2

Solução:

$$\begin{aligned} \text{a)} \quad 11010_2 &= 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 16 + 8 + 0 + 2 + 0 \\ &= 26_{10} \end{aligned}$$

$$\begin{aligned} \text{b)} \quad 1100100_2 &= 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 \\ &= 64 + 32 + 0 + 0 + 4 + 0 + 0 \\ &= 100_{10} \end{aligned}$$

• Conversão Decimal - Binário

Para se converter um número decimal em binário, divide-se sucessivamente o número decimal por 2 (base do sistema binário), até que o último quociente seja 1. Os restos obtidos das divisões e o último quociente compõem um número binário equivalente, como mostra o exemplo a seguir.

Converter os seguintes números decimais em binário.

a) 23_{10}

b) 52_{10}

Solução:

logo:

a) 23

$$23_{10} = 10111_2$$

b) 52

$$52_{10} = 110100_2$$

• Complemento de 2

Na maioria dos sistemas digitais, a operação de subtração é efetuada através da representação de números negativos usando complemento de 2. Por exemplo, a operação $7 - 5$ pode ser representada como sendo $7 + (-5)$. Observe que, na segunda representação, a operação efetuada é uma adição de um número positivo com um negativo.

O complemento de 2 um número binário é obtido adicionando-se 1 ao complemento de 1 do mesmo. O complemento de 1 é obtido simplesmente invertendo-se os dígitos que formam o número.

Exemplo:

Calcule o 2^0 complemento dos seguintes números binários.

a) 1001_2

b) 1101_2

Solução:

a) 1001
 $0110 \rightarrow \text{complemento de 1}$
 $+ \underline{1}$
 $0111 \rightarrow \text{complemento de 2}$

b) 1101
 $0010 \rightarrow \text{complemento de 1}$
 $+ \underline{1}$
 $0011 \rightarrow \text{complemento de 2}$

No exemplo, a o número 9_{10} (1001_2) tem como complemento de 2: 011_2 . O complemento de 2 é a representação negativa do número binário, ou seja, -9_{10} é representado como: 0111_2 .

A subtração binária através do segundo complemento, é realizada somando-se o subtrator com o complemento de 2 do subtraendo, como mostra o exemplo a seguir.

Subtraia os seguintes números em binários.

a) $13_{10} - 7_{10}$

b) $6_{10} - 9_{10}$

Solução:

a) $13_{10} = 1101_2$
 $7_{10} = 0111_2$

Calculando o complemento de 2 de 7_{10} (0111_2), temos:

$$\begin{array}{r} 0111 \\ 1000 \rightarrow \text{complemento de } 1 \\ +1 \\ \hline 1001 \rightarrow \text{complemento de } 2 \end{array}$$

logo:

$$\begin{array}{r} 13 = 1101 \\ -7 = +1001 \\ \hline 6 = 0110 \end{array}$$

OBSERVAÇÃO:

Sempre que houver carry (vai um) do bit mais significativo, ele deverá ser desprezado.

b) $6_{10} = 0110_2$
 $9_{10} = 1001_2$

Calculando o complemento de 2 de 9_{10} (1001_2), temos:

$$\begin{array}{r} 1001 \\ 0110 \rightarrow \text{complemento de } 1 \\ +1 \\ \hline 0111 \rightarrow \text{complemento de } 2 \end{array}$$

Se no resultado da soma (1101) não existe carry (vai um), devemos achar o complemento de 2 deste número e acrescentar o sinal negativo.

$$\begin{array}{r} 1101 \\ 0010 \rightarrow \text{complemento de } 1 \\ +1 \\ \hline 0011 \rightarrow \text{complemento de } 2 \end{array}$$

então:

$$\begin{array}{l} 6 - 9 = -3 \\ \text{ou seja:} \\ -0011 \end{array}$$

OBSERVAÇÃO:

Podemos achar o complemento de 2 de um número binário a partir da seguinte regra: conservamos o primeiro bit 1 (um) menos significativo e efetuamos o complemento de 1 dos bits mais significativos (bits da esquerda)

Sistema de Numeração Hexadecimal

O sistema hexadecimal, ou sistema de base 16, é largamente utilizado nos computadores de todos os portes. Neste sistema são utilizados 16 símbolos para representar cada um dos dígitos hexadecimais, conforme a tabela a seguir:

Nº DECIMAL	DÍGITO HEXADECIMAL	Nº BINÁRIO
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Note que as letras **A, B, C, D, E, F** representam dígitos associados às quantidades **10,11,12,13,14 e 15** respectivamente.

- **Conversão Hexadecimal - Decimal**

Novamente aplicamos para o sistema hexadecimal a definição de um sistema de numeração qualquer. Assim temos:

$$N = dn \cdot 16^n + \dots + d2 \cdot 16^2 + d1 \cdot 16^1 + d0 \cdot 16^0$$

Para se efetuar a conversão, basta adicionar os membros da segunda parcela da igualdade, como ilustra o exemplo a seguir:

$$a) 23_{16}$$

$$b) 3B_{16}$$

Solução:

$$\begin{aligned}a) 23_{16} &= 2 \times 16^1 + 3 \times 16^0 \\23_{16} &= 2 \times 16 + 3 \times 1 \\23_{16} &= 35_{10}\end{aligned}$$

$$\begin{aligned}b) 3B_{16} &= 3 \times 16^1 + 11 \times 16^0 \\3B_{16} &= 3 \times 16 + 11 \times 1 \\3B_{16} &= 59_{10}\end{aligned}$$

Observe que o dígito hexadecimal "B", no exemplo (b), é equivalente ao número 11 decimal, como mostra a tabela apresentada anteriormente.

• Conversão Decimal - Hexadecimal

A conversão decimal-hexadecimal é efetuada através das divisões sucessivas do número decimal por 16, como demonstrado no exemplo a seguir.

$$a) 152_{10}$$

$$b) 249_{10}$$

Solução:

$$a) 152_{10}$$

$$b) 249_{10}$$

Logo:

$$a) 152_{10} = 98_{16}$$

$$b) 249_{10} = F9_{16}$$

Números Decimais Codificados em Binário (BCD)

Como já foi discutido anteriormente, os sistemas digitais em geral, trabalham com números binários. Com o intuito de facilitar a comunicação homem-máquina, foi desenvolvido um código que representa cada dígito decimal por um conjunto de 4 dígitos binários, como mostra a tabela seguinte:

Nº DECIMAL	REPRESENTAÇÃO BINÁRIA
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Este tipo de representação é denominado de **código BCD (Binary-Coded Decimal)**. Desta maneira, cada dígito decimal é representado por grupo de quatro bits, como ilustrado a seguir:

$$527_{10} = 0101\ 0010\ 0111$$

$$527_{10} = 010100100111_2$$

Observe que a conversão decimal-BCD e BCD-decimal é direta, ou seja, separando-se o dígito BCD em grupos de 4 bits, cada grupo representa um dígito decimal, como ilustrado a seguir.

a) 290_{10}

b) 638_{10}

Solução:

$$a) 290_{10} = 0010\ 1001\ 0000$$

$$290_{10} = 001010010000_2$$

b) $638_{10} = 0110\ 0011\ 1000$

$$638_{10} = 011000111000_2$$

Exemplo:

Converter os seguintes números BCD para decimal.

a) 1001010000001000

b) 1001101001

Solução:

a) $1001010000001000 = 1001\ 0100\ 0000\ 1000$

$1001010000001000 = \begin{array}{cccc} 9 & 4 & 0 & 8 \end{array}$

$= 9408_{10}$

b) $001001101001 = 0010\ 0110\ 1001$

$001001101001 = \begin{array}{ccc} 2 & 6 & 9 \end{array}$

$= 269_{10}$

SENAC CLIMATE

SENAC CLIMATEC

ANEXO 2

BIBLIOGRAFIA

- LIMA II, Eduardo. **Uma metodologia para implementação através de CLPs de controle supervisório de células de manufatura utilizando rede de Petri.** 2002. Tese (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia, Universidade Federal da Bahia, Salvador.
- SENAI CETIND. **Apostila de CLP básico.** Lauro de Freitas, s.d.
- SENAI CETIND. **Apostila de CLP Avançado.** Lauro de Freitas, s.d
- GEORGINI, Marcelo. **Automação aplicada descrição e implementação de sistemas seqüenciais com PLCs.** São Paulo: Érica, 2000.
- SILVEIRA, Paulo Rogério da; SANTOS, Wilderson E. **Automação e controle discreto.** São Paulo: Érica, 1998.
- NATALE, Ferdinando. **Automação industrial.** São Paulo: Érica, 2000.
- MIYAGI, Paulo Eigi. Controle programável fundamentos do controle de sistemas a eventos discretos. São Paulo: Edgard Blücher, 1996.
- OLIVEIRA, Júlio César Peixoto de. Controlador programável. São Paulo: Makron Books, 1993.
- BONACORSO, Nelson Gauze. Automação eletropneumática. São Paulo: Érica, 1997.
- ALLEN-BRADLEY . Reference Manual. S.I.p. , s.d.
- SIMPSON, Colin D. Programmable Logic Controllers. S.I.p. 1994.
- Site tutorial sobre CLP na internet: <<http://www.plcs.net>>
- Site da ALLEN-BRADLEY na internet: <<http://www.ab.com>>
- Site da SMAR na internet: <<http://www.smar.com.br>>
- Site da ALTUS na internet: <<http://www.altus.com.br>>
- Site da SIEMENS na internet: <<http://www.siemens.com>>
- Site da MODICON na internet: <<http://www.modicon.com>>
- Site da Editora Érica na internet: <<http://www.erica.com.br>>