

**ESCOLA DE EDUCAÇÃO PROFISSIONAL SENAI “VISCONDE DE MAUÁ”
CURSO TÉCNICO DE INFORMÁTICA INDUSTRIAL**

CONTROLADORES LÓGICOS PROGRAMÁVEIS SIEMENS SIMATIC S7-200



LABORATÓRIO DE PROJETOS E AUTOMAÇÃO

MAIO DE 2003

SUMÁRIO

	INTRODUÇÃO	3
1	INTRODUÇÃO AO CLP	3
1.1	HISTÓRICO.....	3
1.2	MERCADO ATUAL.....	4
1.3	APLICAÇÕES.....	5
2	PRINCIPIOS DE FUNCIONAMENTO	6
2.1	ESTRUTURAS DO CLP.....	6
2.2	HARDWARE CARACTERÍSTICO.....	7
3	INTRODUÇÃO A PROGRAMAÇÃO	9
3.1	LÓGICA MATEMÁTICA E BINÁRIA.....	9
3.2	LINGUAGENS DE PROGRAMAÇÃO.....	11
4	O CLP SIEMENS SIMATIC S7-200	13
4.1	CARACTERÍSTICAS DE HARDWARE.....	14
4.1.1	ESPECIFICAÇÕES TÉCNICAS.....	15
4.2	CARACTERÍSTICAS DO SOFTWARE.....	16
5	INSTALANDO O CLP S7-200	17
5.1	INSTALANDO EM PAINEL OU TRILHO.....	18
5.2	LIGANDO O CLP AO COMPUTADOR.....	18
5.3	INSTALANDO O STEP 7 Micro/WIN 32.....	19
6	PROGRAMANDO O CLP S7-200	20
6.1	CONCEITOS BÁSICOS.....	20
6.2	CONFIGURANDO O CLP.....	22
6.3	PROGRAMANDO EM LADDER.....	23
6.4	INSTRUÇÕES LADDER.....	26
6.5	EXEMPLOS DAS INSTRUÇÕES LADDER.....	44
7	EXERCÍCIOS	64

1 – INTRODUÇÃO AO CLP

Os Controladores Lógicos Programáveis ou CLPs, são equipamentos eletrônicos utilizados em sistemas de automação flexível. São ferramentas de trabalho muito úteis e versáteis para aplicações em sistemas de acionamentos e controle, e por isso são utilizados em grande escala no mercado industrial. Permitem desenvolver e alterar facilmente a lógica para acionamento das saídas em função das entradas. Desta forma, podemos associar diversos sinais de entrada para controlar diversos atuadores ligados nos pontos de saída.

1.1 – HISTÓRICO

Durante a década de 50, os dispositivos eletromecânicos foram os recursos mais utilizados para efetuar controles lógicos e de intertravamentos nas linhas de produção e em máquinas isoladas. Tais dispositivos são baseados principalmente em relés, tinham especial importância na indústria automobilística em que a complexidade dos processos produtivos envolvidos exigia, não raro, instalações em painéis e cabinas de controle com centenas de relés e, conseqüentemente, um número maior ainda de interconexões deles. Tais sistemas de controle, apesar de funcionais, apresentavam problemas de ordem prática bastante relevantes. Como as instalações possuíam uma grande quantidade de elementos, a ocorrência de uma falha qualquer significava o comprometimento de várias horas ou mesmo dias de trabalho de pesquisa e correção do elemento faltoso. Além disto, pelo fato de os relés apresentarem dimensão física elevada, os painéis ocupavam grande espaço, o qual deveria ser protegida contra umidade, sobre temperatura, gases inflamáveis, oxidações, poeira, etc.

Outro fator ainda comprometedor das instalações a relés era o fato de que como a programação lógica do processo controlado era realizada por interconexão elétrica com lógica fixas (hardwired), eventuais alterações na mesma exigiam interrupções no processo produtivo a fim de se reconectarem os elementos. Interrupções estas nem sempre bem-vindas na produção industrial. Como conseqüência ainda, tornava-se obrigatória a atualização das listas de fiação como garantia de manter a documentação do sistema.

Com o advento da tecnologia de estado sólido, desenvolvida a princípio em substituição às válvulas a vácuo, alguns dispositivos transistorizados foram utilizados no final da década de 50 e início dos anos 60, sendo que tais dispositivos reduziam muitos dos problemas existentes nos relés. Porém, foi com o surgimento dos componentes eletrônicos integrados em larga escala (LSI), que novas fronteiras se abriam ao mundo dos computadores digitais e, em especial às tecnologias para a automação industrial.

Assim, a primeira experiência de um controle de lógica que permitisse a programação por recursos de software foi realizada em 1968, na divisão de hidramáticos da General Motors Corporation. Aliado ao uso de dispositivos periféricos, capazes de realizar operações de entrada e saída, um minicomputador com sua capacidade de programação pode obter vantagens técnicas de controle que suplantaram o custo que tal implementação representou na época. Iniciava-se a era dos controladores de lógica programável.

Essa primeira geração de PLC, como poderia ser denominada, recebeu sensíveis melhorias com o advento dos microprocessadores ocorrido durante os anos 70. Assim, não se tornava necessário o uso de computadores de grande porte, tornando-o uma unidade isolada. Foram adicionados ainda recursos importantes tais como interfaces de

operação e programação facilitadas ao usuário, instruções de aritmética e de manipulação de dados poderosas, recursos de comunicação por meio de redes de PLC, possibilidades de configuração específica a cada finalidade por meio de módulos intercambiáveis, dentre outras inúmeras vantagens encontradas nos modelos comerciais que estão atualmente disponíveis.

Assim, os técnicos em controle de máquinas e processos passaram a contar com um dispositivo capaz de:

- Permitir fácil diagnóstico de funcionamento ainda na fase de projeto do sistema e/ou de reparos em falhas que venham a ocorrer durante a sua operação.
- Ser instalado em cabinas reduzidas devido ao pequeno espaço físico exigido.
- Operar com reduzido grau de proteção, pelo fato de não serem gerados faiscamentos.
- Ser facilmente reprogramado sem necessidade de interromper o processo produtivo (programação on-line).
- Possibilitar a criação de um banco de armazenamento de programas que podem ser reutilizados a qualquer momento.
- Manter uma documentação sempre atualizada com o processo em execução.
- Apresentar baixo consumo de energia.
- Manter o funcionamento da planta de produção com uma reduzida equipe de manutenção.
- Garantir maior confiabilidade pela menor incidência de defeitos.
- Emitir menores níveis de ruídos eletrostáticos.
- Ter a flexibilidade de expansão do número de entradas e saídas por serem controladas.
- Ter a capacidade de se comunicar com diversos outros equipamentos

Em nível de Brasil, porém, é na década de 80, que o PLC veio a proliferar na indústria, primeiramente pela absorção de tecnologias utilizadas na Matriz das multinacionais. Atualmente, com a crescente redução no custo do PLC, observa-se o incremento de sua utilização nas indústrias em geral, independente de seu porte ou ramo de atividades.

Nota: Citação de Silveira, Paulo Rogério e Santos, Winderson em Automação e controle Discreto.

1.2 – MERCADO ATUAL

A roda viva da atualização, da qual fazemos parte, movimenta e impulsiona o mercado mundial atualmente. Os profissionais buscam conhecimentos para se tornarem mais versáteis, adequando-se às necessidades das empresas, que por sua vez, buscam maior variedade e rapidez de produção para atender ao cliente, que se torna cada vez mais exigente.

As empresas estão se reorganizando para atender as necessidades atuais de aumento de produtividade, flexibilidade e redução de custos. Destas necessidades surgiram as necessidades de os equipamentos se adequarem rapidamente às alterações de configurações necessárias para produzirem diversos modelos de produtos, com pequenas alterações entre si.

1.3 – APLICAÇÕES

Projetado para substituir antigos quadros de comando de relés o controlador deve ocupar pequeno espaço físico, apresentar flexibilidade para possíveis mudanças na lógica de controle, ser resistente ao ambiente e ser imune a toda natureza de ruídos.

O CLP trabalha manipulando saídas conforme o estado de suas entradas. O usuário elabora um programa, normalmente por software que dá os resultados desejados. CLPs são muito usados hoje. São boas as chances que haja um CLP presente em vários tipos de indústrias. Se você trabalha em máquinas automatizadas ou em processos industriais você provavelmente está os usando. Quase qualquer aplicação que precisa de algum tipo de controle pode se usar um CLP. Algumas aplicações de sucesso são:

- Máquinas industriais em geral
- Prensas
- Máquinas de usinagem de madeira
- Aparafusadeiras
- Manipuladores
- Máquinas de solda
- Estações de tratamento de efluentes
- Estações de bombeamento de fluidos
- Elevadores
- Transportadores de cargas
- Máquinas de lavar veículos
- Sistemas de empacotamento
- Automação predial
- Sistema de alarme com monitoramento remoto

2 – PRINCÍPIOS DE FUNCIONAMENTO

Neste capítulo serão apresentados os princípios técnicos de funcionamento de um CLP. As características físicas e técnicas são comuns em alguns aspectos. O que ocorre no mercado, são as características específicas que um ou outro fabricante adiciona ao seu clp, como por exemplo, facilidade de programação, conectores padronizados, tamanhos reduzidos, potências de controle, etc.

2.1 – ESTRUTURAS DO CLP

Podemos apresentar a estrutura de um CLP dividida em três partes: entrada, processamento e saída.



Figura 1 – Estrutura básica de um CLP

Os sinais de entrada e saída dos CLPs podem ser digitais ou analógicos. Existem diversos tipos de módulos de entrada e saída que se adequam as necessidades do sistema a ser controlado.

Os módulos de entrada e saídas são compostos de grupos de bits, associados em conjunto de 8 bits (1 byte) ou conjunto de 16 bits, de acordo com o tipo da CPU.

As entradas analógicas são módulos conversores A/D, que convertem um sinal de entrada em um valor digital, normalmente de 12 bits (4096 combinações). As saídas analógicas são módulos conversores D/A, ou seja, um valor binário é transformado em um sinal analógico.

Os sinais dos sensores são aplicados às entradas do controlador e a cada ciclo (varredura) todos esses sinais são lidos e transferidos para a unidade de memória interna denominada memória imagem de entrada. Estes sinais são associados entre si e aos sinais internos. Ao término do ciclo de varredura, os resultados são transferidos à memória imagem de saída e então aplicados aos terminais de saída. Este ciclo está representado na figura 2.2.



Figura 2 – Ciclo de processamento dos CLPs.

2.2 – HARDWARE CARACTERÍSTICO

Basicamente o CLP é um computador dedicado a funções de controle, este possui um processador, memória e dispositivos de entrada e saída como veremos mais adiante. Um sistema de controle de estado sólido, com memória programável para armazenamento de instruções para controle lógico, pode executar funções equivalentes as de um painel de relés ou de um sistema de controle lógico. É ideal para aplicações em sistemas de controle de relés e contadores, os quais se utilizam principalmente de fiação, dificultando, desta forma, os acessos, possíveis modificações e ampliações do circuito de controle existente.

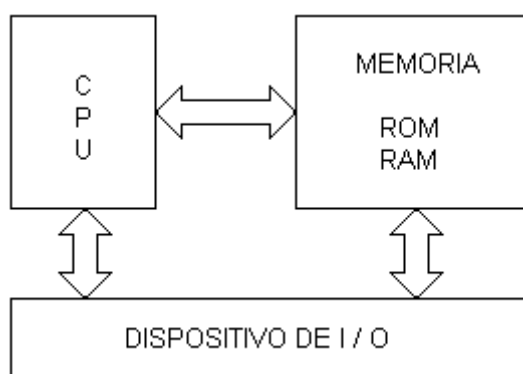


Figura 3 – Esquema de hardware característico.

O Controlador Programável monitora o estado das entradas e saídas, em resposta às instruções programadas na memória do usuário, e energiza ou desenergiza as saídas, dependendo do resultado lógico conseguido através das instruções de programa.

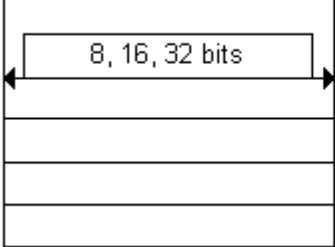
O programa é uma sequência de instruções a serem executadas pelo Controlador Programável para executar um processo. A tarefa do Controlador Programável é ler, de forma cíclica, as instruções contidas neste programa, interpretá-las e processar as operações correspondentes.

Um CLP é basicamente um pequeno computador dedicado, onde em sua estrutura físico encontra-se:

- Unidade Central de Processamento
- Memória do tipo ROM para armazenamento do FIRMWARE (programa onde se encontra os principais códigos de operação da máquina)
- Memória RAM para armazenamento de dados e programas do usuário
- Dispositivos de Entrada e Saída para a comunicação com o exterior.

Por outro lado, algumas características são particulares nos CLPs, como por exemplo:

- Espaço de memória RAM com mapeamento para uso específico na aplicação fim, em outras palavras durante o projeto do controlador lógico programável seus espaços de memória são previamente organizados durante a elaboração do FIRMWARE. Isto ocorre porque os CLPs são equipamentos dedicados a um tipo de aplicação específica, admitindo apenas serem programados com SOFTWARES desenvolvidos especificamente para eles

	ENDEREÇO DAS PALAVRAS DE MEMÓRIA		
	DECIMAL	OCTAL	HEXADECIMAL
	255	377	FF
	511	777	1FF
	4095	7777	FFF

- Normalmente é utilizado um dispositivo de SOFTWARE que monitora o tempo limite para a varredura do programa do usuário (watch dog time).
- Os dispositivos de entrada saída (pontos digitais), são geralmente isolados para evitar ruídos e também a danificação interna por picos de tensão na entrada ou saída.

O processador do CP efetua a leitura das entradas e atualiza a tabela imagem de entrada, logo após executa o programa do usuário e atualiza a tabela imagem de saída.

3 – INTRODUÇÃO A PROGRAMAÇÃO

Dentro deste capítulo, veremos a base lógica de um CLP e de onde surgiram as linguagens de programação, bem como as diversas linguagens existentes no mercado e as mais usadas. Para dominarmos uma linguagem de programação, devemos primeiramente entender como funciona a lógica matemática e binária, muito usada na lógica digital.

O cuidado dedicado ao desenvolvimento de uma linguagem adequada é prioridade do fabricante. Desta forma, o formato torna-se fundamental para aceitação do produto no mercado. Este deve ser de fácil operação e também oferecer uma grande flexibilidade de programação e estruturação.

3.1 – LÓGICA MATEMÁTICA E BINÁRIA

A lógica matemática ou simbólica visa superar as dificuldades e ambigüidades de qualquer língua, devido a sua natureza vaga e equívoca das palavras usadas e do estilo metafórico e, portanto, confuso que poderia atrapalhar o rigor lógico do raciocínio. Para evitar essas dificuldades, criou-se uma linguagem lógica artificial.

A lógica binária possui apenas dois valores que são representados por : 0 e 1. A partir desses dois símbolos construímos então uma base numérica binária. A partir desses conceitos foram criadas as portas lógicas, que são circuitos utilizados para combinar níveis lógicos digitais de formas específicas. Neste curso aprenderemos apenas as portas lógicas básicas: AND, OR e NOT.

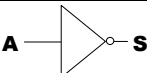
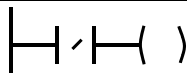
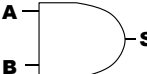
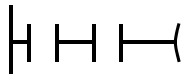

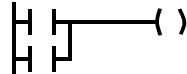
Portas Lógicas	Símbolo	Expressão	Ladder
NOT		$S = \overline{A}$	
AND		$S = A.B$	
OR		$S = A+B$	

Tabela 1 – Portas lógicas e suas equivalências em Ladder.

Os CLPs vieram a substituir elementos e componentes eletro-eletrônicos de acionamento e a linguagem utilizada na sua programação é similar à linguagem de diagramas lógicos de acionamento desenvolvidos por eletrotécnicos e profissionais da área de controle, esta linguagem é denominada linguagem de contatos ou simplesmente LADDER. A linguagem Ladder permite que se desenvolvam lógicas combinacionais, seqüenciais e circuitos que envolvam ambas, utilizando como operadores para estas lógicas: entradas, saídas, estados auxiliares e registros numéricos. A Tabela 2 nos mostra os três principais símbolos de programação:

Tipo	Símbolo	Equipamento elétrico
Contato Aberto		
Contato Fechado		
Saída		

Tabela 2 – Principais símbolos de programação.

Para entendermos o circuito com o CLP, vamos observar o programa desenvolvido para acender a lâmpada L quando acionamos o botão B1.

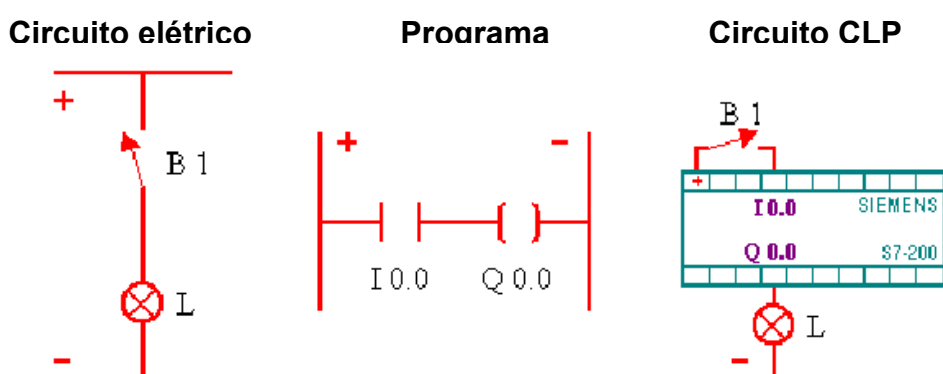


Figura 4 – Acionamento de uma lâmpada.

O botão B1, normalmente aberto, está ligado a entrada I0.0 e a lâmpada está ligada à saída Q0.0. Ao acionarmos B1, I0.0 é acionado e a saída Q0.0 é energizada. Caso quiséssemos que a lâmpada apagasse quando acionássemos B1 bastaria trocar o contato normal aberto por um contato normal fechado, o que representa a função NOT. Podemos desenvolver programas para CLPs que correspondam a operações lógicas combinacionais básicas da álgebra de Boole, como a operação AND. Na área elétrica a operação AND corresponde a associação em série de contatos, como indicado na figura X.

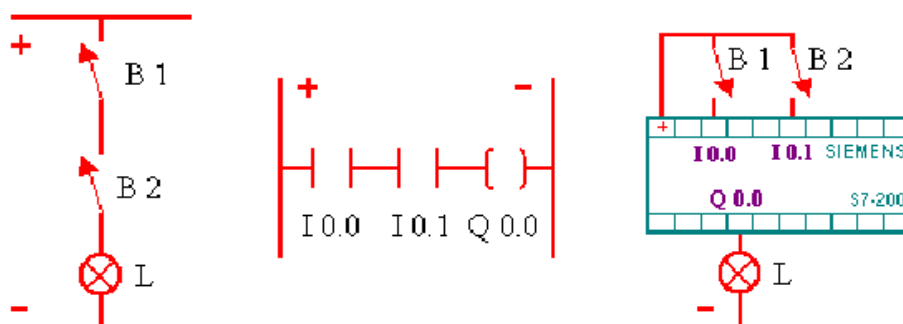


Figura 5 – Lógica AND.

Outra operação lógica básica é a função OR, que corresponde a associação em paralelo de contatos, como indicado na figura X.

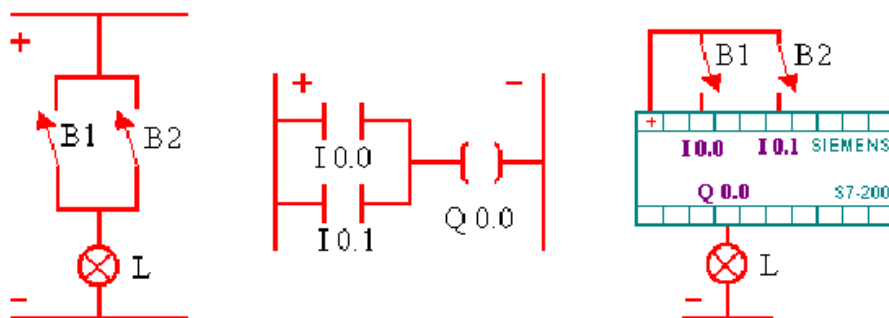


Figura 6 – Lógica OR.

Assim podemos afirmar que todas as funções lógicas combinacionais podem ser desenvolvidas em programação e executadas por CLPs, uma vez que todas derivam dos básicos: NOT, AND e OR.

A flexibilidade dos CLPs é percebida neste momento pois as alterações lógicas podem ocorrer com grande facilidade, sem que sejam necessárias alterações do hardware ou inclusão de componentes eletrônicos ou elétricos. Esta é a principal característica dos sistemas de automação flexíveis e o que faz dos CLPs ferramentas de grande aplicação nas estruturas de automação.

Além da linguagem de contatos, existem outras formas de programação características de cada fabricante.

Concluimos então que os projetos de automação e controle envolvendo CLPs reduzem o trabalho de desenvolvimento de hardware dos circuitos lógicos do acionamento, bem como os dispositivos e potência para acionamento de cargas e dos atuadores, uma vez que podemos escolher módulos de saída já prontos, adequados ao tipo de carga que desejamos acionar.

As utilizações desses controladores contemplam, por conseguinte alguns passos genéricos:

- Definição da função lógica a ser programada
- Transformação desta função em programa assimilável pelo CLP
- Implementação física do controlador e de suas interfaces com o processo

3.2 – LINGUAGENS DE PROGRAMAÇÃO

Entre os formatos encontrados no mercado o mais popular é o LADDER. Dentre outros tipos existentes no mercado temos: LISTA DE INSTRUÇÕES, BLOCOS LÓGICOS, LINGUAGENS DESCRITIVAS, etc

O CLP SIEMENS SIMATIC S7-200 e o software STEP 7 MICRO/WIN, suportam os seguintes formatos de programação:

- Statement List (STL) é um conjunto de instruções escritas pelo usuário, onde cada linha de programação representa uma função do CLP. Essa linguagem geralmente é usada por programadores experientes. O STL permite que o programador crie programas que não possam ser feitos em linguagem visual.
- Ladder (LAD) é bastante conhecido e teve seu nascimento no continente norte americano. É também conhecido como “Diagrama de contatos”, muito semelhante a um esquema elétrico. Por se tratar de uma linguagem visual,

de fácil compreensão e aprendizado, neste manual todos os exercício e exemplos serão referidos a ela.

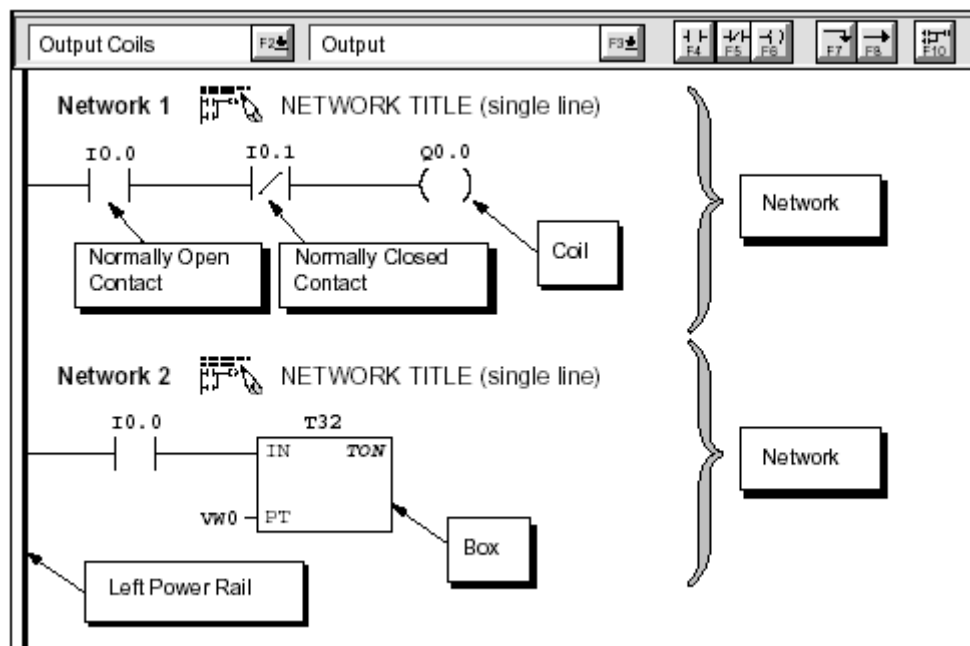


Figura 7 – Elementos básicos de uma lógica Ladder.

```
LD    I0.0    //Read one input
A     I0.1    //AND with another input
=     Q1.0    //Write value to output 1
```

Figura 8 – Estrutura de um programa STL (Lista de Instruções).

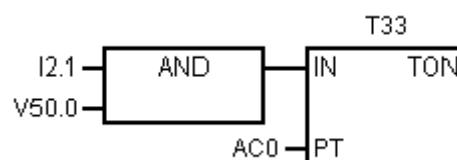


Figura 9 – Programa em FBD (Diagrama de Blocos).

4 – O CLP SIEMENS SIMATIC S7-200

O micro CLP S7-200 constitui uma verdadeira alternativa econômica para todas as aplicações na área de automação de pequeno porte. Seu projeto é caracterizado pelas seguintes qualidades básicas:

- Elevada performance
- Excelente modularidade
- Elevada conectividade

Elevada performance

O S7-200 é pequeno e compacto – ideal para as aplicações onde o espaço disponível é crítico. Ele também é rápido, oferecendo um excelente comportamento em tempo real, garantindo maior qualidade, eficiência e confiabilidade ao processo. E, com seus recursos amigáveis de programação, ele pode ser programado de maneira rápida, simples e conveniente.

Excelente modularidade

A família do S7-200 tem uma concepção modular coerente, permitindo que soluções possam ser desenvolvidas sob medida e ampliadas conforme a demanda. Ela é composta de CPUs com diferentes níveis de memória e diferentes números de entradas e saídas integradas. Estão disponíveis uma vasta gama de módulos de expansão para diversas funções, bem como diversas possibilidades de painéis de comando e visualização.

Elevada conectividade

As possibilidades de comunicação do S7-200 não têm comparação. As interfaces integradas padrão RS485 suportam taxas de transferência de dados até 187,5 Kbps e podem trabalhar no modo Freeport, que aceita protocolo definido pelo usuário. Através de módulos de expansão específicos, é possível a comunicação via modem, PROFIBUS-DP, AS-Interface e até Ethernet.

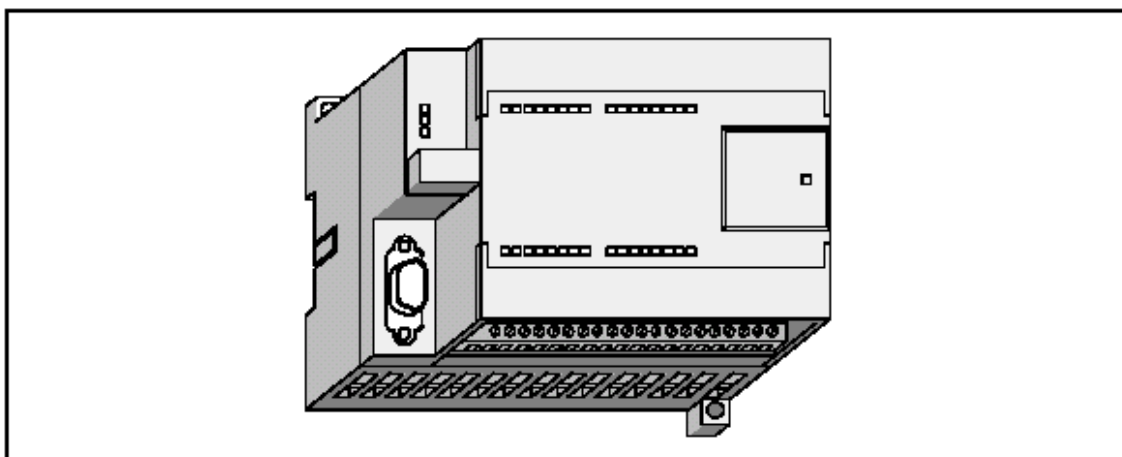


Figura 10 – S7-200 Micro PLC

4.1 – CARACTERÍSTICAS DE HARDWARE

Um amplo espectro de CPUs está disponível para aplicações simples até aplicações de grande performance. Ao todo são 5 modelos de CPU com diferentes características, dentre elas, a quantidade de memória e de entradas e saídas integradas. A própria CPU já vem equipada com diversos recursos, como:

- Entradas e saídas digitais integradas
- Interface RS485 integrada
- Protocolo PPI (mestre/escravo), MPI (escravo), ou outros como Modbus (programável via Freeport)
- Contadores rápidos
- Saídas de pulso rápido
- Memória retentiva
- Entradas de interrupção
- Relógio de tempo real (opcional para alguns modelos)
- Cartão de memória removível
- Potenciômetro

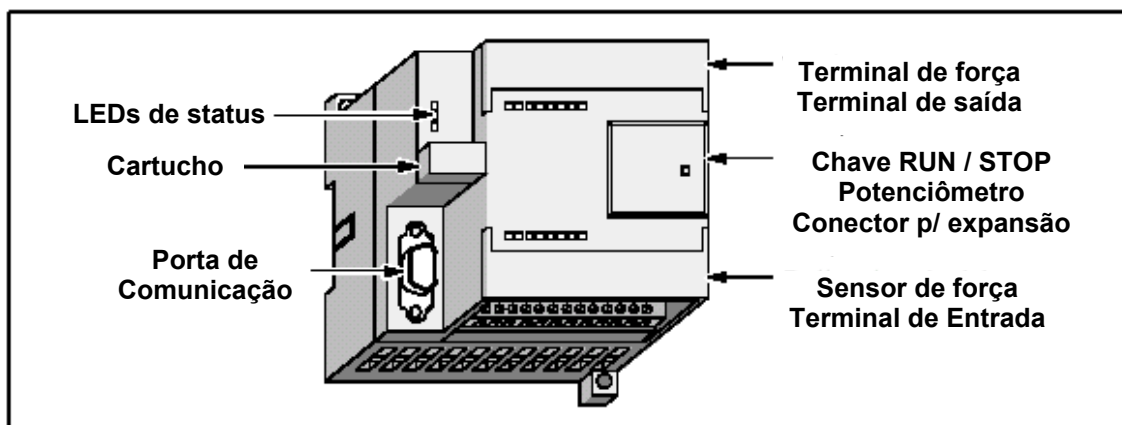


Figura 11 – Características do S7-200.

A grande diversidade de módulos de expansão permite a adaptação da configuração para diversos tipos de aplicação. Dependendo do modelo da CPU, é possível utilizar até 7 módulos de expansão:

Módulos de Entradas/Saídas

- Digitais
- Analógicos
- Específicos para medição de temperatura

Módulo de Posicionamento

Módulos de Comunicação

- AS-Interface (mestre)
- PROFIBUS-DP (escravo)
- Modem
- Ethernet

4.1.1 – ESPECIFICAÇÕES TÉCNICAS

Abaixo, a tabela com as especificações técnicas dos CLPs SIMATIC S7-200 modelos 221, 222 e 224:

Feature	CPU 221	CPU 222	CPU 224
Physical Size of Unit	90 mm x 80 mm x 62 mm	90 mm x 80 mm x 62 mm	120.5 mm x 80 mm x 62 mm
Memory			
Program	2048 words	2048 words	4096 words
User data	1024 words	1024 words	2560 words
Memory type	EEPROM	EEPROM	EEPROM
Memory cartridge	EEPROM	EEPROM	EEPROM
Data backup (super capacitor)	50 hours typical	50 hours typical	190 hours typical
Local I/O			
Local I/O	6 In/4 Out	8 In/6 Out	14 In/10 Out
Number of expansion modules	none	2 modules	7 modules
Total I/O			
Digital I/O image size	256 (128 In/128 Out)	256 (128 In/128 Out)	256 (128 In/128 Out)
Digital I/O physical size	10	62	128
Analog I/O image size	none	16 In/16 Out	16 In/16 Out
Analog I/O physical size	none	12 In/10 Out	12 In/10 Out
Instructions			
Boolean execution speed	0.37 µs/instruction	0.37 µs/instruction	0.37 µs/instruction
Internal relays	256	256	256
Counters/Timers	256/256	256/256	256/256
Sequential control relays	256	256	256
For/Next loops	Yes	Yes	Yes
Integer math (+ - * /)	Yes	Yes	Yes
Real math (+ - * /)	Yes	Yes	Yes
Enhanced Features			
Built-in high-speed counter	4 (20 KHz)	4 (20 KHz)	6 (20 KHz)
Analog adjustments	1	1	2
Pulse outputs	2 (20 KHz, DC only)	2 (20 KHz, DC only)	2 (20 KHz, DC only)
Communication interrupts	1 transmit/2 receive	1 transmit/2 receive	1 transmit/2 receive
Timed interrupts	2 (1 ms to 255 ms)	2 (1 ms to 255 ms)	2 (1 ms to 255 ms)
Hardware input interrupts	4	4	4
Real-time clock	Yes (cartridge)	Yes (cartridge)	Yes (built-in)
Password protection	Yes	Yes	Yes
Communications			
Number of communication ports:	1 (RS-485)	1 (RS-485)	1 (RS-485)
Protocols supported Port 0:	PPI, MPI slave, Freeport	PPI, MPI slave, Freeport	PPI, MPI slave, Freeport
PROFIBUS peer-to-peer	(NETR/NETW)	(NETR/NETW)	(NETR/NETW)

Tabela 3 – Especificações técnicas

4.2 – CARACTERÍSTICAS DO SOFTWARE

A programação do SIMATIC S7-200 é feita através do software STEP 7-Micro/WIN, que é uma ferramenta que preza a facilidade de uso, possibilitando a programação na linguagem que mais lhe agrada: LAD, FDB e STL (SIMATIC), ou KOP e FUP (IEC 1131). Sua aparência e operação são idênticas às aplicações padrão Windows, agilizando a ambientação do usuário. Ele permite que você crie suas próprias bibliotecas, com partes de programas para serem reutilizadas, ou adicionar bibliotecas prontas, como a de protocolo USS. Além disso, ele conta com os Wizards, que são assistentes de parametrização para funções como comunicação em rede e configuração do TD200, entre outros. Isso tudo lhe permite poupar tempo, aumentando sua produtividade e reduzindo custos.

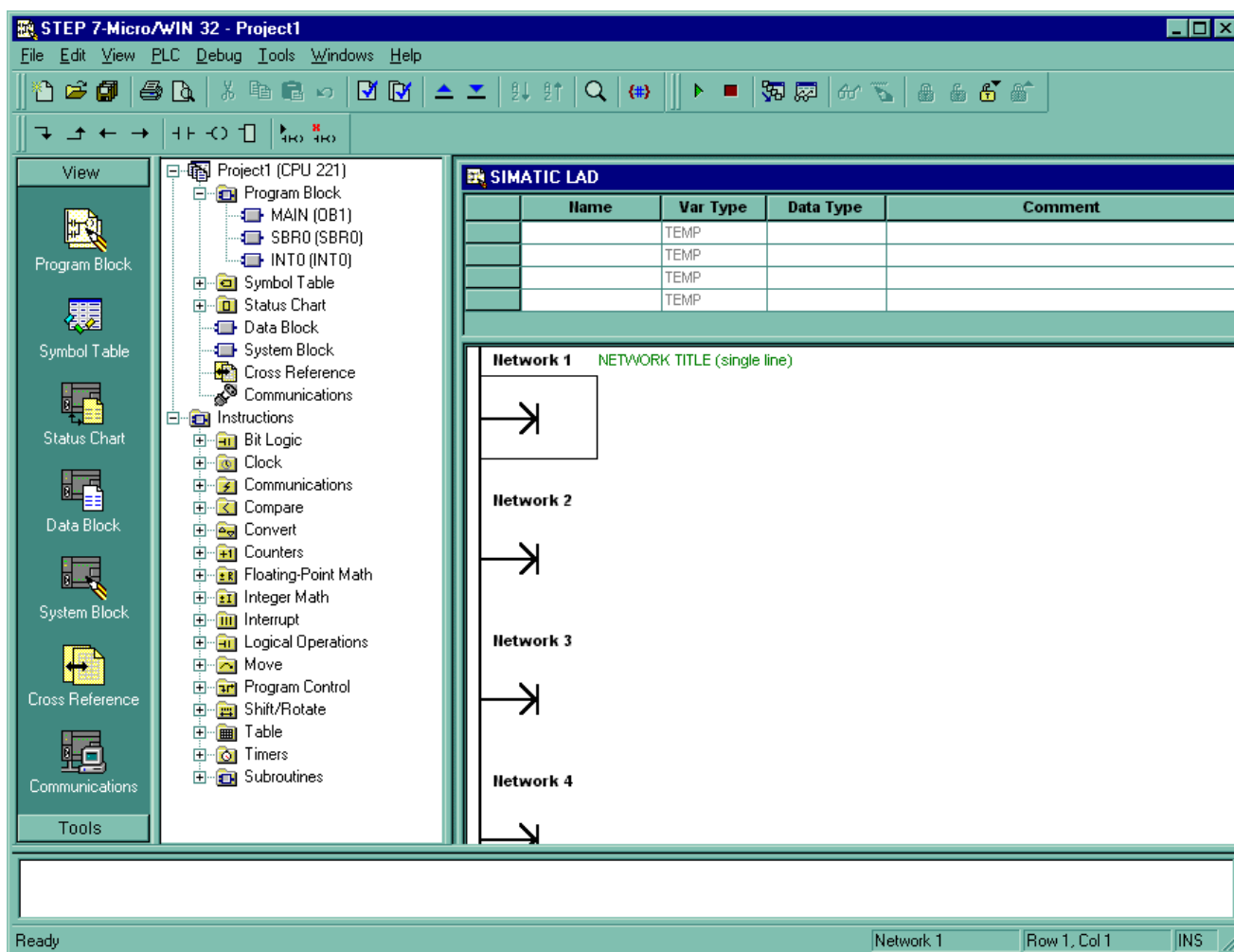


Figura 12 – STEP 7 Micro/WIN 32

5 - INSTALANDO O CLP S7-200

A instalação de um CLP S7-200 foi desenvolvida para ser fácil, rápida e prática. Você pode usar os furos da carcaça para prender o CLP em um painel ou então usar os clips para fixá-lo em um trilho padrão(DIN). O pequeno tamanho do CLP facilita o aproveitamento de espaço em um painel.

Você pode instalar o clp em posição vertical ou horizontal, contanto que os módulos de expansão sejam instalados da mesma maneira. Para montagens em trilho padrão(DIN) existe um cabo para a interligação dos módulos de expansão.

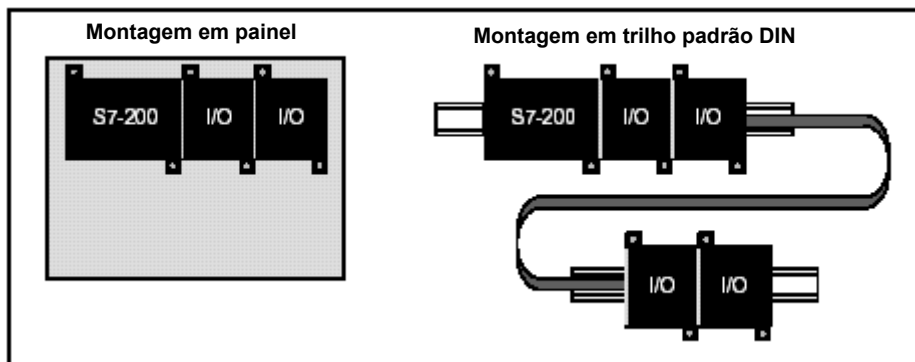


Figura 13 – Configurações de montagem.

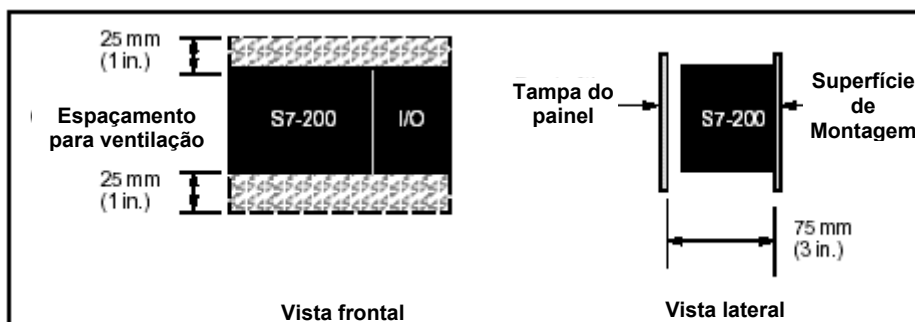


Figura 14 – Espaçamento mínimo vertical e horizontal para montagem

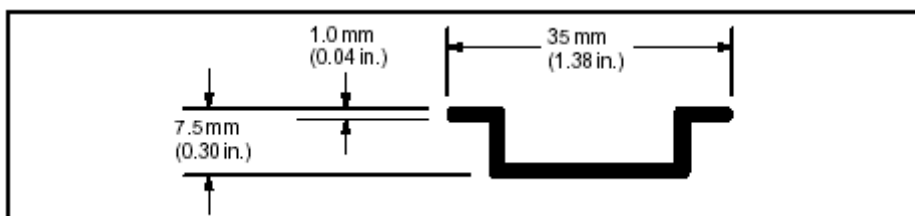


Figura 15 – Dimensões para trilho padrão DIN

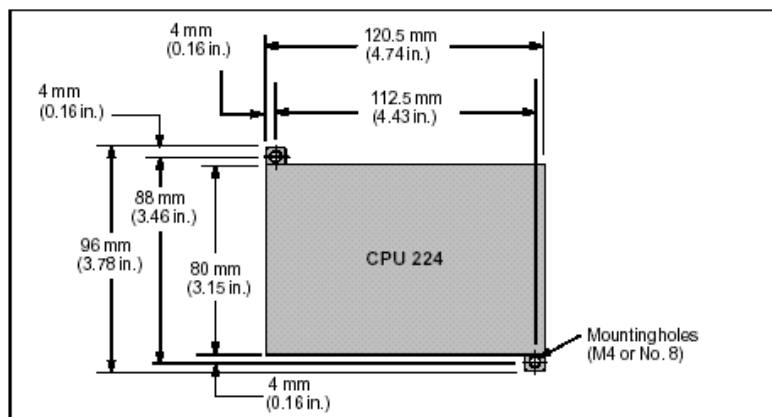


Figura 16 – Dimensões para montagem em painel (somente CPU 224)

5.1 – INSTALANDO EM PAINEL OU TRILHO

Instalando em painel:

- Faça a furação do painel segundo as medidas citadas anteriormente. Os furos são norma DIN M4 ou AMERICAN STANDART NUMBER 8 SCREWS.
- Parafuse o CLP usando os parafusos com as medidas dos furos.
- Nunca manuseie o CLP ligado. Você pode danificá-lo ou tomar choques elétricos.

Instalando em trilho:

- A distância entre cada trilho deve ser de 4" verticalmente.
- Abra o clip localizado na parte de baixo, e encoste o clip no trilho;
- Feche o clip e certifique-se de que o clip esteja bem firme ao trilho.

5.2 – LIGANDO O S7-200 AO COMPUTADOR

A ligação do CLP ao computador se faz através de um cabo PC/PPI, próprio desse CLP, pela porta serial do computador (COM1). A configuração desse cabo será explicada nos próximos capítulos deste manual.

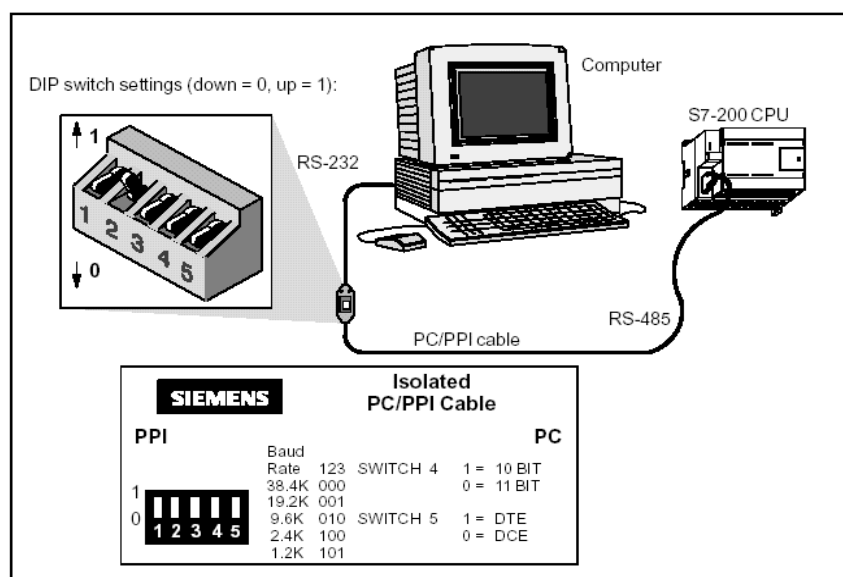


Figura 17 – Ligação através do cabo PC/PPI

O CLP deve ser alimentado por uma fonte de tensão DC de 24V. A alimentação deve ser conectada nas entradas L+ (positivo) e M (terra) conforme a figura abaixo:

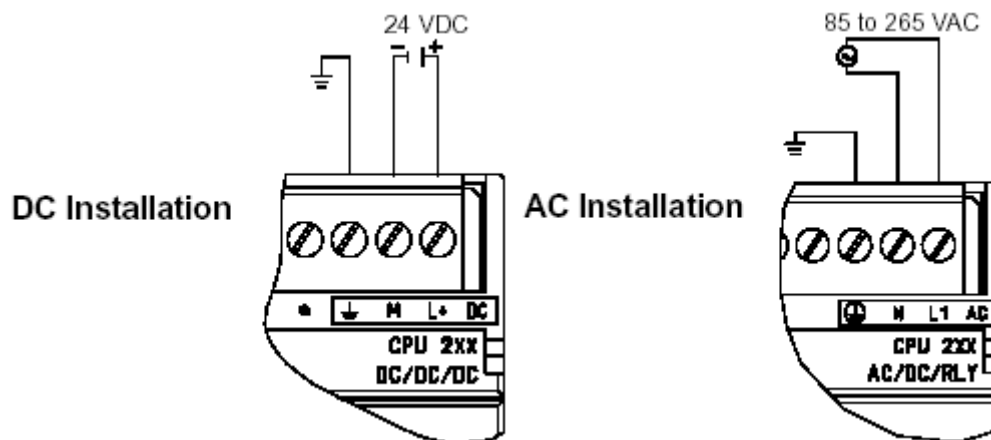


Figura 18 – Alimentação do S7-200.

5.3 – INSTALANDO O STEP 7 Micro/WIN 32

A instalação do programador do CLP S7-200 é simples. Basta ter um microcomputador com os seguintes requerimentos mínimos:

- Sistema Operacional Windows 95, 98, ME, 2000 ou NT 4.0
- Processador de 233MHz
- 32MB RAM
- 50MB de espaço livre em disco.

Basta colocar o CD1 e executar o programa SETUP.EXE dentro do diretório raiz do CD.

6 – PROGRAMANDO O CLP S7-200

A programação do CLP SIEMENS S7-200 será apresentada de uma forma rápida, visual e direta. A linguagem de programação escolhida, foi a LADDER por sua fácil compreensão e programação.

6.1 – CONCEITOS BÁSICOS

ENTRADAS E SAÍDAS

O CLP SIEMENS SIMATIC S7-200 CPU 224 apresenta 14 entradas e 10 saídas reais digitais.

Entradas são elementos usados para monitorar um evento como uma botoeira, interruptor, termostato, etc. As entradas são representadas da seguinte forma:

I0.0, I0.1, I0.2, I0.3, I0.4, I0.5, I0.6, I0.7, I1.0, I1.1, I1.2, I1.3, I1.4, I1.5.

Saídas são elementos usados para acionar equipamentos como motores, válvulas, relés, solenóides, etc. As saídas são representadas da seguinte forma:

Q0.0, Q0.1, Q0.2, Q0.3, Q0.4, Q0.5, Q0.6, Q0.7, Q1.0, Q1.1.

Entradas e saídas virtuais são aquelas apenas representadas internamente pelo programa do CLP. Elas são representadas por:

Ix.y e Qx.y, onde x e y variam de 0 à 7, sendo que a primeira entrada virtual começa em I1.6 e a primeira saída em Q1.2.

BARRAS DE FERRAMENTAS E MENUS DO STEP 7

Ao abrir o STEP 7, aparecerá a tela de abertura do programa. A partir daí, pode-se iniciar a programação de sua lógica LADDER. Abaixo segue a descrição de cada função das barras de ferramentas e menus do STEP 7:

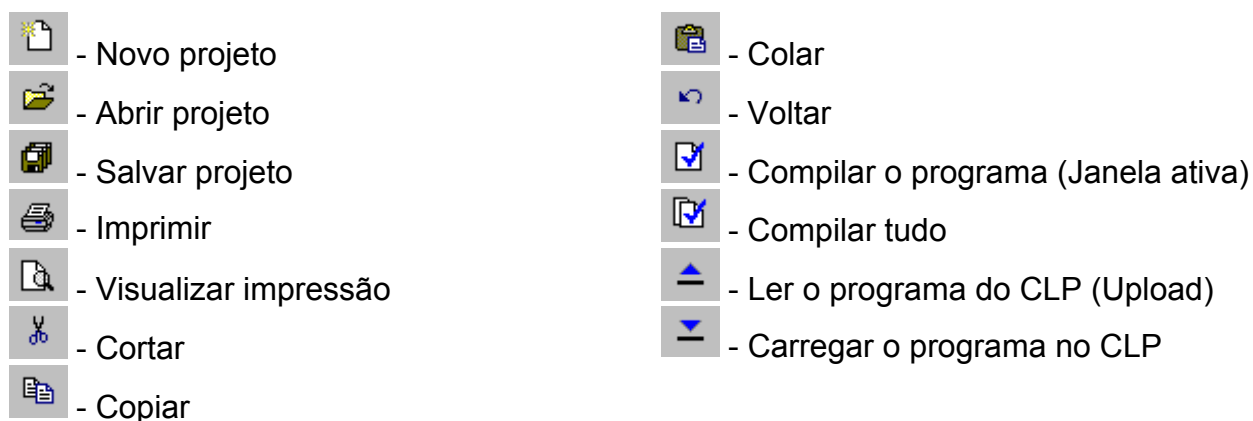


Figura 19 – Barra de ferramentas FILE.

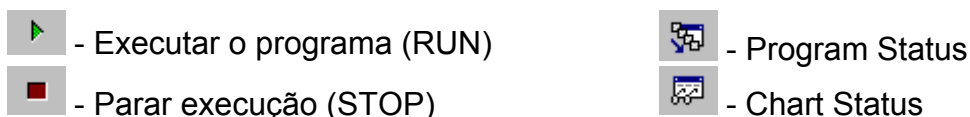


Figura 20 – Barra de ferramentas DEBUG.

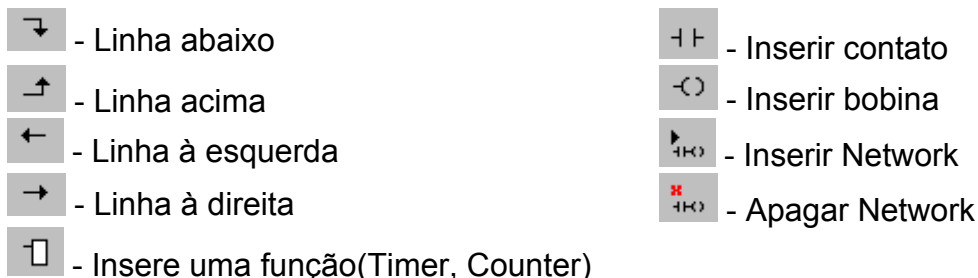


Figura 21 – Barra de ferramentas PROGRAM.



Figura 22 – MENU FILE E EDIT.

O menu VIEW contém os modos de visualização da linguagem de programação (STL, LADDER e FBD), A tabela de símbolos, o mapa de status, o *Data Block* e o *System Block*, referencia cruzada e configura o tipo de CLP em *Communications*.

Em PLC, rodamos o programa compilamos, limpamos a lógica, *Reset*, e escolhemos o tipo de CLP em *Type*.

O menu FILE e EDIT possui todos os comandos básicos do windows, com o qual já estamos familiarizados.

O comando *Upload* e *Download* é o mesmo que o citado na barra de ferramentas. *Import* e *Export* abre e salva o programa em código ASCII. *Insert* e *Delete* são responsáveis pela inclusão e exclusão de Networks, Subroutines e Interrupts.

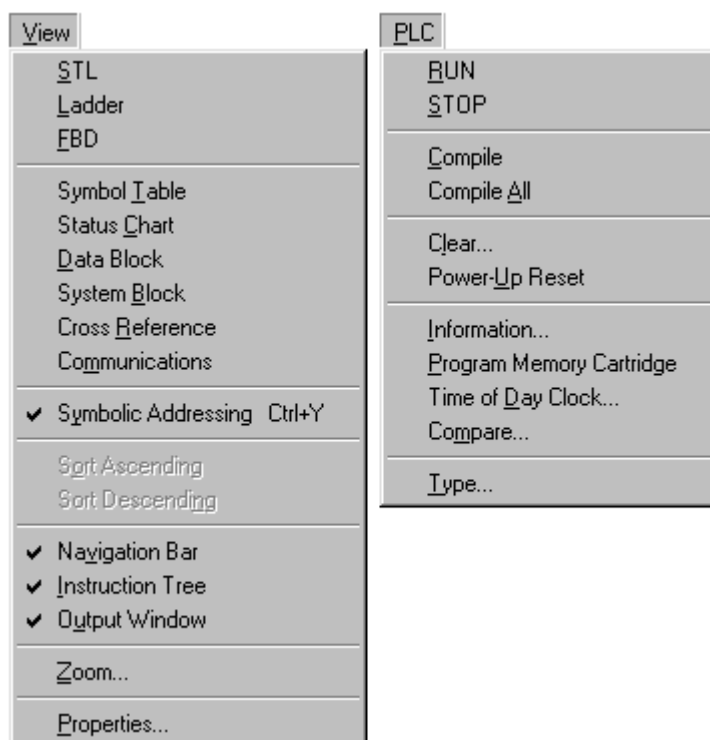
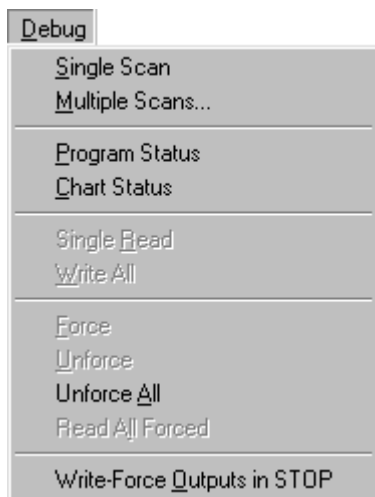


Figura 23 – MENU VIEW E PLC



Este menu serve para executar scans simples e múltiplos. Em *Program Status* monitoramos o programa em execução pela tela do computador.

MENUS TOOLS, WINDOWS E HELP

O menu *Tools* serve para configurarmos as barras de ferramentas (*Customize...*), as opções de programação do CLP (*Options...*). Há também um assistente de programação.

O menu *Windows* seleciona a janela que desejamos visualizar e o menu *Help*, contém todos os tópicos de ajuda para utilizar o programa.

Figura 24 – MENU DEBUG

6.2 – CONFIGURANDO O CLP

Para configurarmos o tipo de CLP a ser usado devemos seguir os seguintes passos:

- Pelo ícone **Communications**:

Clique no ícone ao lado
ou pelo menu
View/Communications

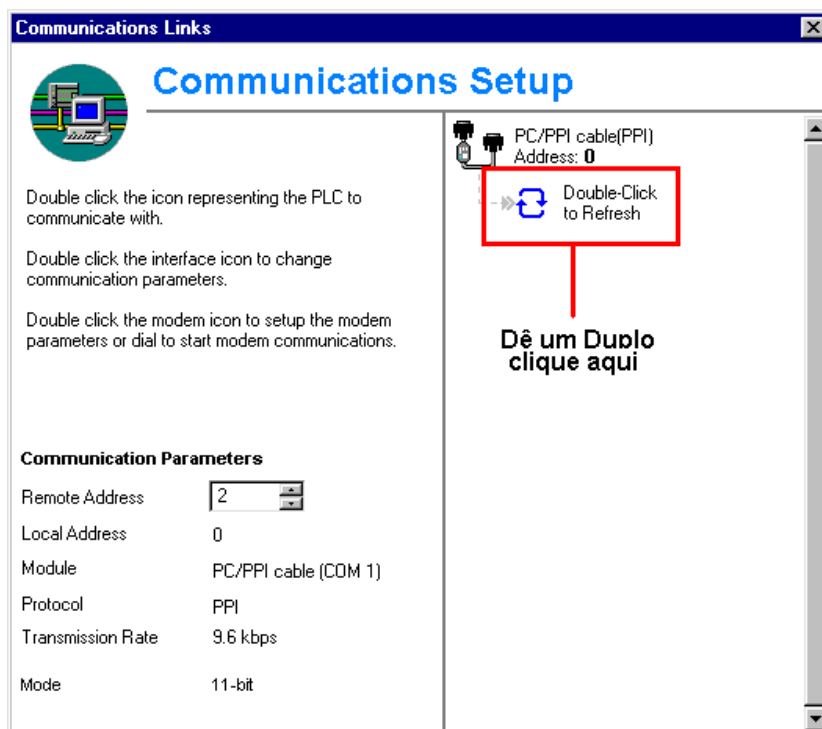


Figura 25 – Configurando o CLP e sua comunicação

- Pelo menu **PLC/Type**, aparecerá a seguinte tela:

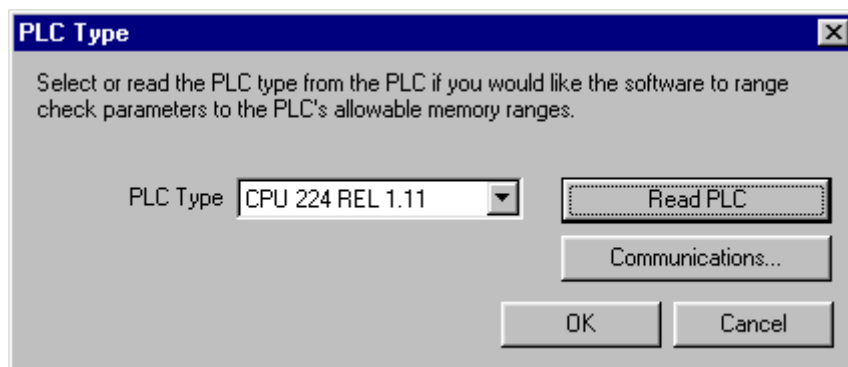


Figura 26 – Escolhendo o tipo de CLP.

Clique em **Read PLC** (com o CLP já conectado a porta serial e ligado) e de **OK**

6.3 – PROGRAMANDO EM LADDER

A primeira coisa a se fazer antes de programar, é escolher a linguagem de programação. Abra o menu *View* e selecione *Ladder*. Depois é só seguir os passos abaixo:

1. Para inserir um contato, coloque o cursor no início da Network 1. Clique em ou pressione F4. Aparecerá uma drop down para escolher o tipo de contato.

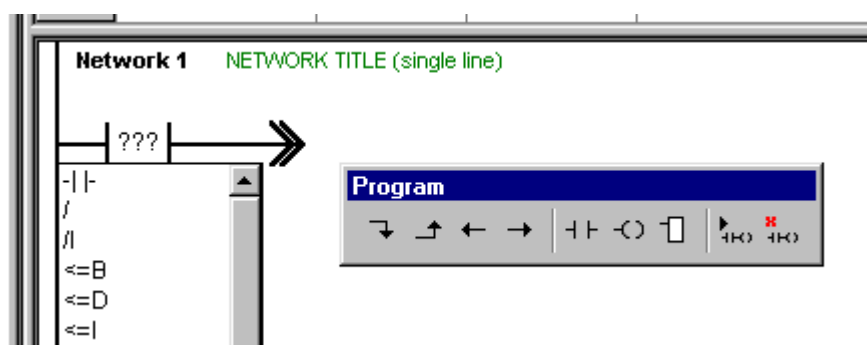


Figura 27 – Inserindo um contato,

Para inserir um contato normalmente aberto selecione . Para um normalmente fechado selecione .

2. Indique o endereço do contato (Já citados no capítulo 6.3, em ENTRADAS E SAÍDAS)

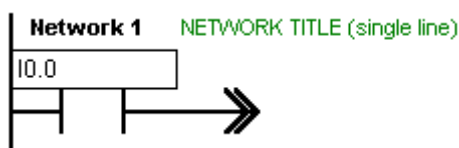



Figura 28 – Nome ao contato

Você pode dar um nome a Network, indicando qual sua função, e adicionar comentários não visíveis na lógica LADDER.

3. Insira uma saída clicando em  ou pressione F6. Aparecerá uma drop down para escolher o tipo de saída.

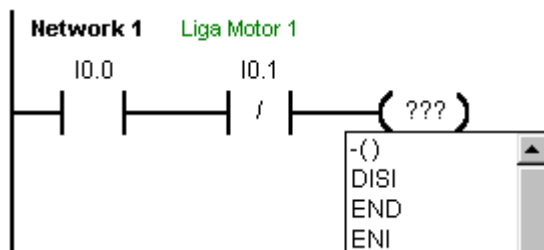


Figura 29 – inserindo uma saída.

4. Indique o endereço da saída (já citadas no capítulo 6.3, em ENTRADAS E SAÍDAS)

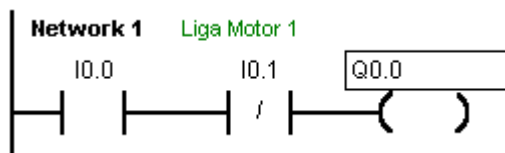


Figura 30 – Nomeando uma saída.

5. O programa esta pronto. Basta agora compilar e enviar ao CLP.

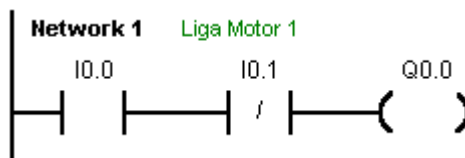




Figura 31 – Programa pronto.

Para compilar o Program Editor, System Block, e o Data Block clique em  (Compile All). Para enviar o programa ao CLP clique em  (Download). Surge a seguinte tela:

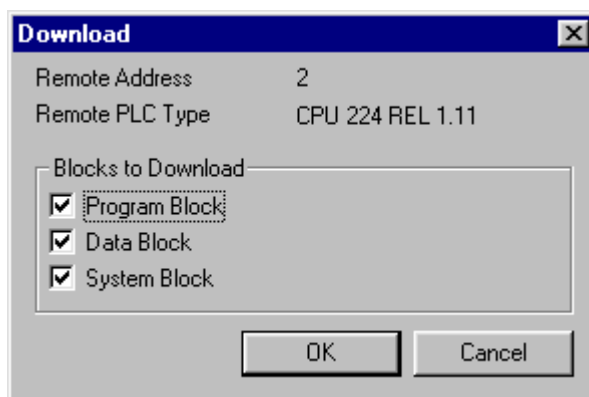




Figura 32 – Enviando o programa ao CLP.

De OK, aguarde o programa gravar no CLP e depois execute o programa clicando em  (RUN). Para monitorar pela tela do micro clique em  (Program Status).

Está pronto o primeiro passo para programar um CLP SIEMENS S7-200. Não esqueça antes de gravar o programa no CLP de colocar a chave RUN/STOP em TERM.(Figura 11).

Existe uma outra maneira de programar o CLP, usando a *Instruction Tree*. Basta selecionar uma lógica e arrastá-la para a edição LADEER(LAD):

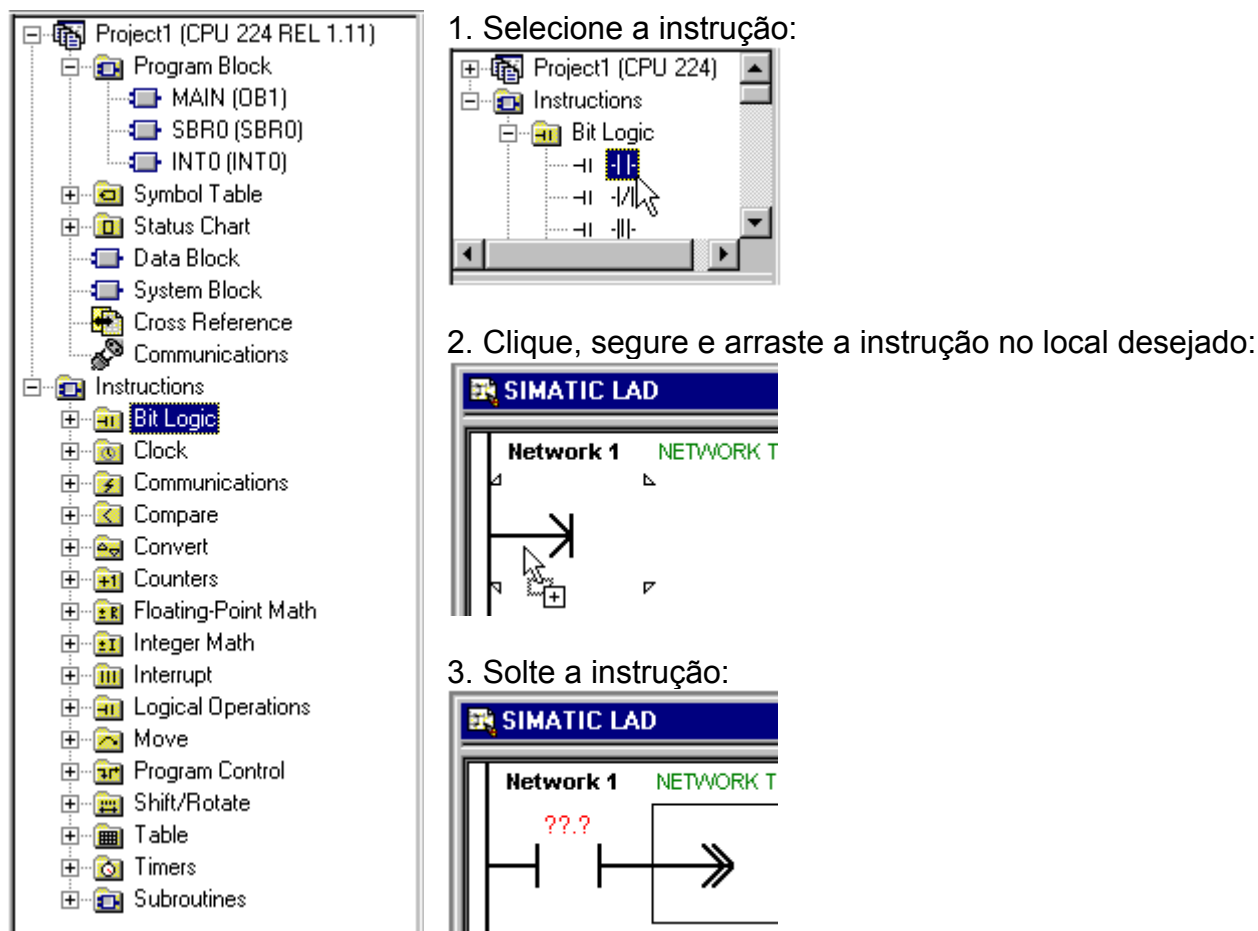


Figura 33 – Instruction Tree

Não esqueça de colocar os nomes das respectivas entradas e saídas. Basta dar um clique sobre os pontos de interrogação em vermelho.

Você pode também, ao invés de arrastar a instrução para o editor LAD, posicionar o cursor onde gostaria de inserir uma instrução e dar um duplo clique na instrução desejada da *Instruction Tree*.

6.4 – TABELA DE SÍMBOLOS

A tabela de símbolos serve para tornar o nosso programa mais amigável e fácil de manusear. Com ela, podemos atribuir nomes que nós compreendemos a entradas e saídas, agilizando a localização das mesmas. Para visualizar a tabela de símbolos basta clicar no botão Symbol Table da barra VIEW, ou então ir pelo menu **View / Symbol Table** ou ainda clicar na **Instruction Tree** em Symbol Table e escolher a tabela corrente:

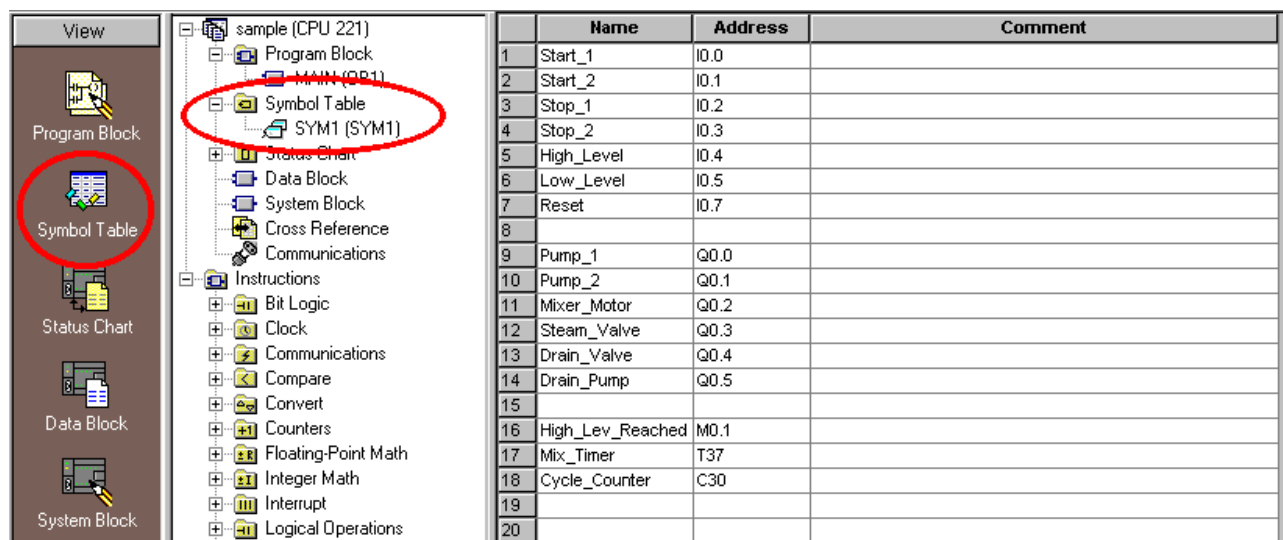


Figura 34 – Tabela de Símbolos.

Podemos criar várias tabelas de acordo com a necessidade do programa. Para inserir uma tabela devemos ir pelo menu **Edit / Insert / Table**. Ou então clicar com o botão do mouse em cima do ícone da Symbol Table na Instruction Tree e selecionar **Insert Symbol Table**:

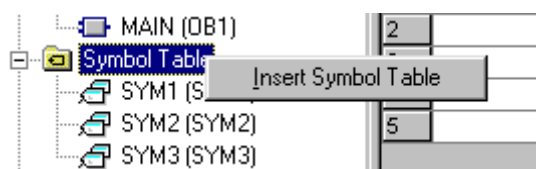


Figura 35 – Inserindo uma tabela de símbolos.

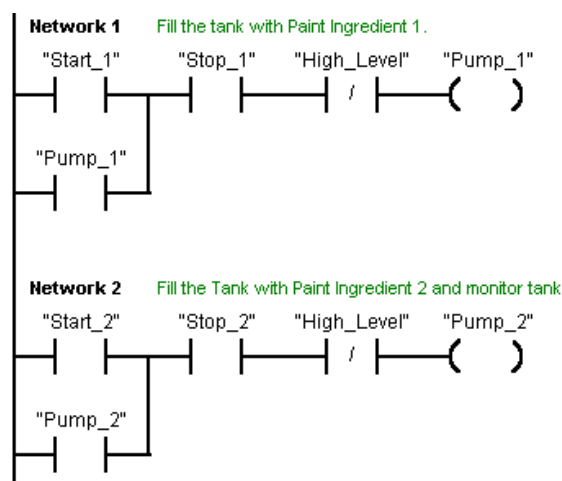


Figura 36 – Programa usando Símbolos

6.5 – CONFIGURANDO O SYSTEM BLOCK

Para configurar o System Block basta acessar o menu **View / System Block**, o então clicar em System Block na instruction Tree ou na Barra View. Abra as seguintes janelas:

The screenshot shows the 'System Block' dialog box with the 'Port(s)' tab selected. The 'Port 0' column is active, showing configuration for Port 0. The 'Port 1' column is empty. A 'Defaults' button is visible on the right. The configuration parameters are as follows:

Parameter	Port 0	Port 1	Range
PLC Address	2		(range 1 .. 126)
Highest Address	126		(range 1 .. 126)
Baud Rate	9.6 kbps		
Retry Count	3		(range 0 .. 8)
Gap Update Factor	1		(range 1 .. 100)

Configuration parameters must be downloaded before they take effect.

Not all PLC types support every System Block option. Press F1 to see which options are supported by each PLC.

Buttons: OK, Cancel

Na orelha Ports configuramos o endereço do CLP, o Baud Rate, o Retry Count (Tentativas de contagem), e o Gap Update Factor (Fator de Atualização com erros)

The screenshot shows the 'System Block' dialog box with the 'Retentive Ranges' tab selected. The 'Data Area', 'Offset', and 'Number of Elements' columns are visible. The 'Range 0' is highlighted. The configuration parameters are as follows:

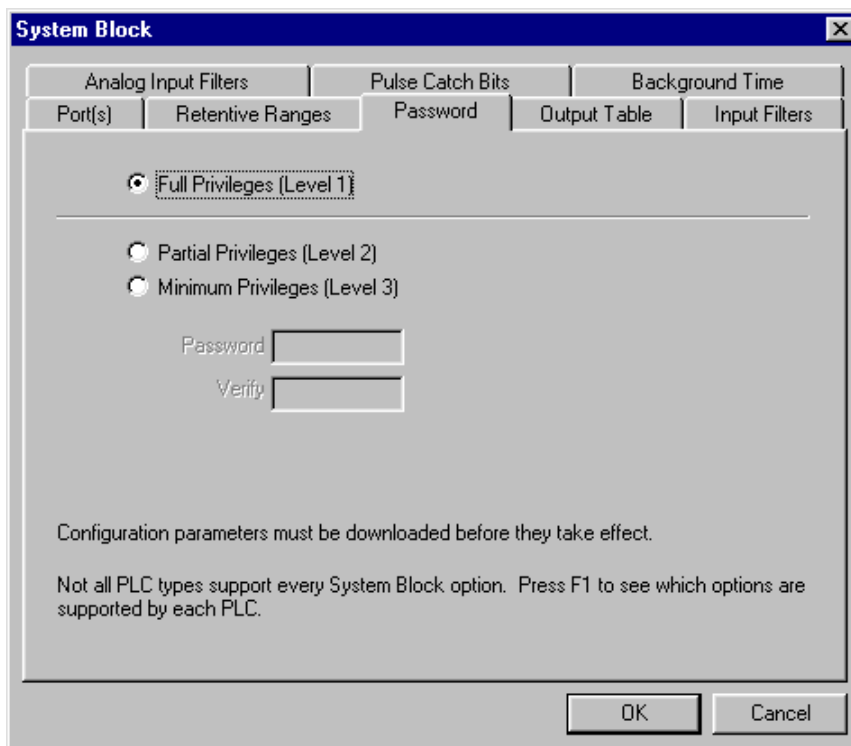
Range	Data Area	Offset	Number of Elements	Action
Range 0	VB	0	2048	Clear
Range 1	VB	0	0	Clear
Range 2	T	0	32	Clear
Range 3	T	64	32	Clear
Range 4	C	0	256	Clear
Range 5	MB	14	18	Clear

Configuration parameters must be downloaded before they take effect.

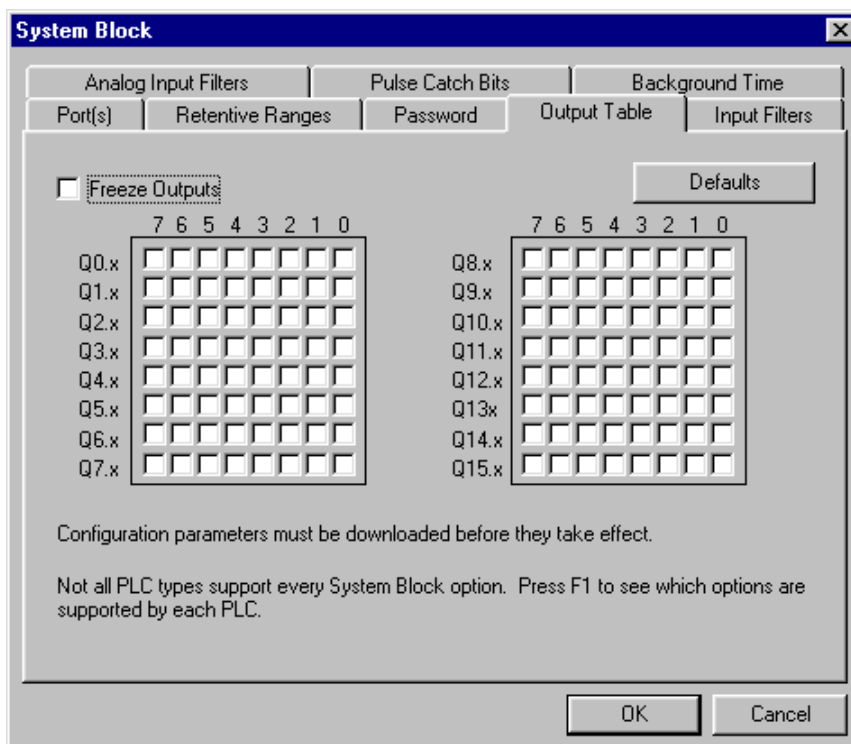
Not all PLC types support every System Block option. Press F1 to see which options are supported by each PLC.

Buttons: OK, Cancel

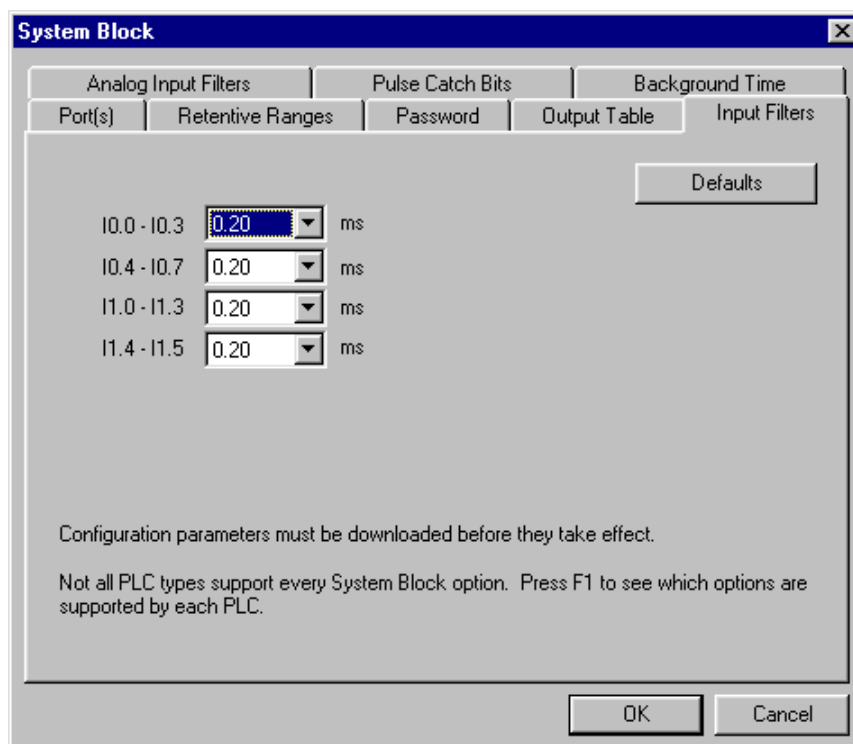
Selecione da orelha Retentive Ranges (alcances de retenção), quais as memórias deverão segurar o valor quando o CLP for desligado.



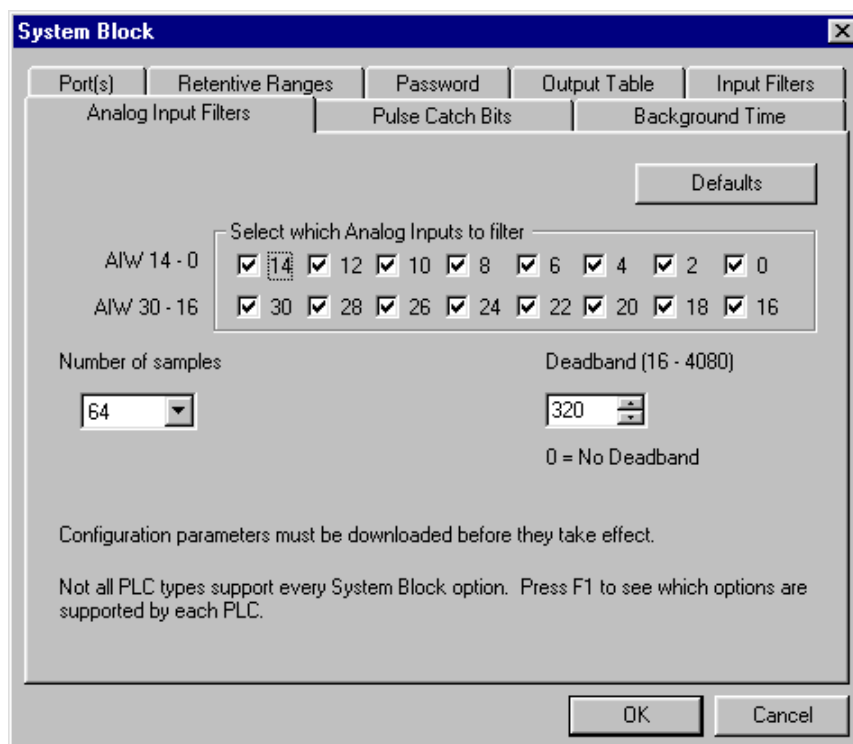
Em Password, podemos definir uma senha de acesso a certas funções do programa, como Download, Upload, modificar o programa etc.



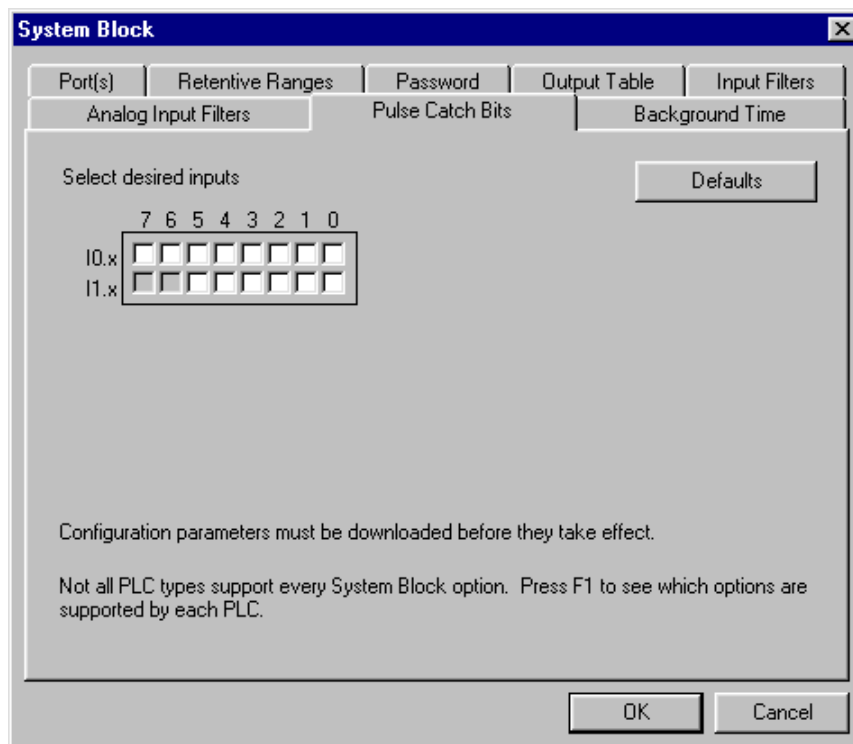
Em Output Table, definimos quais as saídas queremos que mantenham seus estados entre a transição de RUN e STOP do CLP.



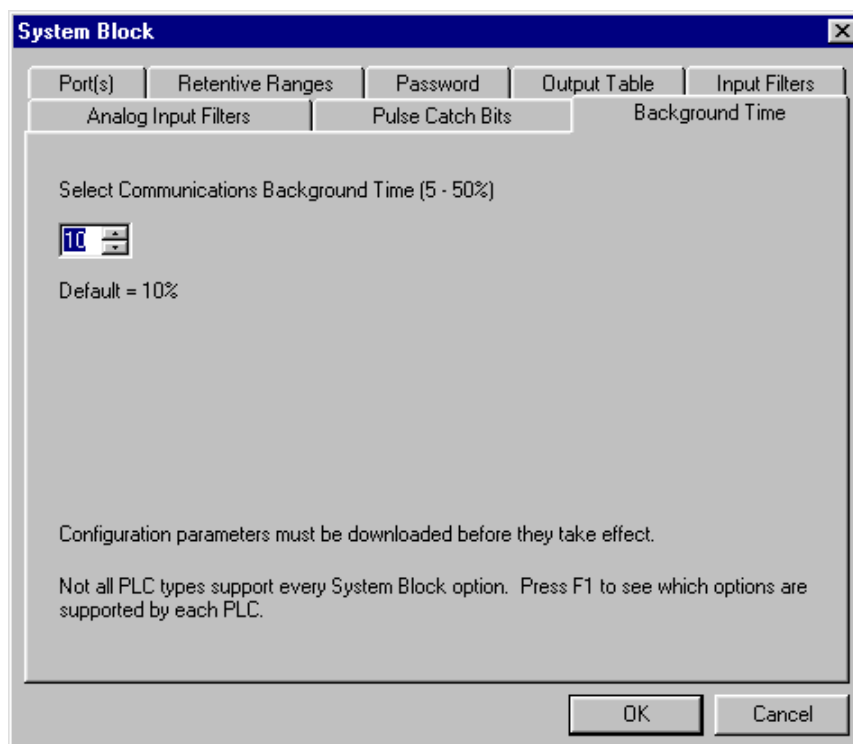
Os Input Filters servem para filtrar as entradas digitais, evitando ruídos externos. Para que essa entrada acione, o novo estado de entrada dela deve durar o mesmo tempo setado em milissegundos.



Os filtros de entrada analógica são usados para evitar ruídos e interferências no sinal analógico.



A configuração Pulse Catch Bits é usada para ler pulsos nas entradas no momento em que elas são atualizadas pela varredura do programa (Scan Cycle).

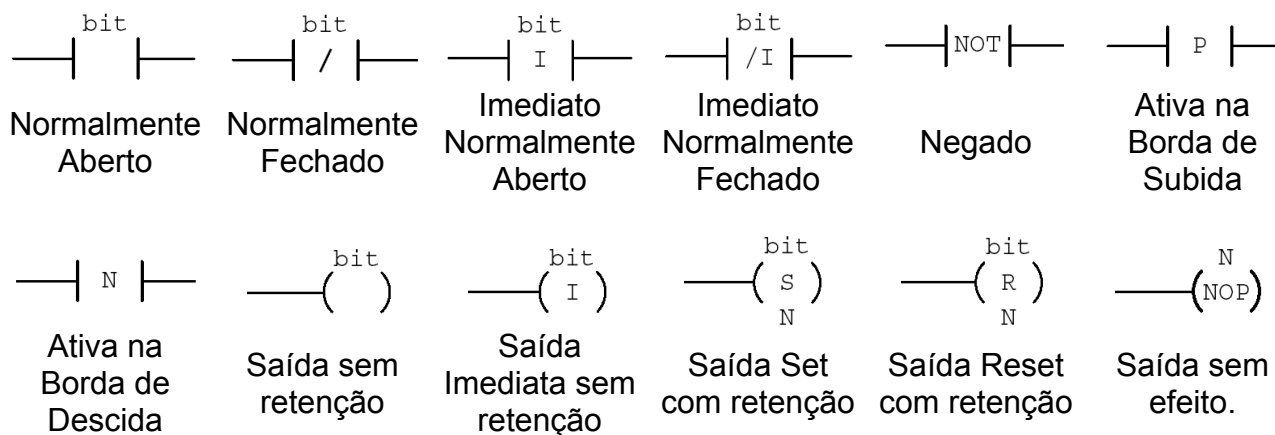


Você pode configurar a porcentagem do tempo de ciclo da varredura que será dedicado a processar os pedidos de uma comunicação associados com o status de STL.

6.4 – INSTRUÇÕES LADDER

Abaixo, segue uma lista das principais instruções de programação da linguagem SIMATIC LAD, ou seja, o ladder usado pelo CLP SIEMENS SIMATIC S7-200.

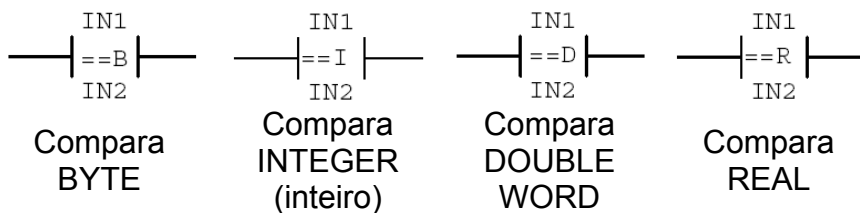
INSTRUÇÕES DE LÓGICA BIT (BIT LOGIC INSTRUCTIONS)



Os contatos normais pegam o valor referenciado na memória ou no Registro do processo da imagem se o tipo de dado é I ou Q. Os contatos imediatos obtêm o valor físico da entrada quando a instrução é executada, mas o Registro do processo da imagem não é renovado. O contato NOT nega a saída de um contato qualquer. O contato P deixa passar apenas um pulso quando ativo em borda de subida e o contato N em borda de descida. A saída NOP serve apenas para gastar ciclos de máquina. O alcance de pontos N das saídas SET e RESET é de 0 até 255.

Inputs / Outputs	Operands	Data Types
	Normalmente Aberto / Fechado	
bit (LAD/STL)	I, Q, M, SM, T, C, V, S, L	BOOL
	Contatos imediatos	
bit (LAD/STL)	I	BOOL
	Saída comum	
bit	I, Q, M, SM, T, C, V, S, L	BOOL
Input (LAD)	Power Flow	BOOL
	Saída Set / Reset / NOP	
bit	I, Q, M, SM, T, C, V, S, L	BOOL
N	VB, IB, QB, MB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	BYTE

Tabela 4 – Operandos de entradas e saídas

INSTRUÇÕES DE COMPARAÇÃO (COMPARE INSTRUCTIONS)

Comparações INTEGER são sinalizadas ($16\#7FFF > 16\#8000$).

Comparações DOUBLE WORD são sinalizadas ($16\#7FFFFFFF > 16\#80000000$).

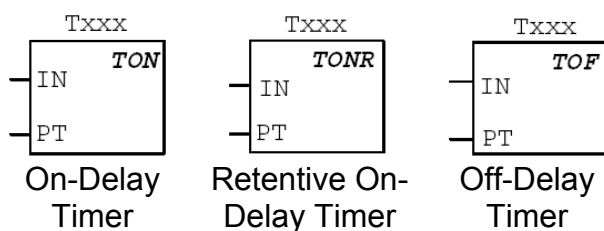
Comparações REAL são sinalizadas

As instruções de comparação servem para comparar dois valores: $IN1$ e $IN2$. As comparações incluem: $IN1=IN2$, $IN1>=IN2$, $IN1<=IN2$, $IN1>IN2$, $IN1<IN2$, $IN1<>IN2$. Elas são ativas (on) quando verdadeiras.

Inputs / Outputs	Operands	Data Types
inputs	BYTE	
	IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD	BYTE
	INTEGER	
	IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC Constant, *VD, *AC, *LD	INT
	DOUBLE WORD / REAL	
	ID, QD, MD, SD, SMD, VD, LD, HC, AC, Constant, *VD, *AC, *LD	DINT/REAL

Tabela 5 – Operandos de comparações

TIMER



O On-Delay Timer e o Retentive On-Delay Timer começam a contar o tempo quando a entrada IN é ligada (ativa). Quando o valor (Txxx) é igual ou maior ao valor do preset (PT) o timer bit é ligado (ativo). O On-delay Timer é zerado quando sua entrada IN é desligada. Já o Retentive On-Delay Timer não é zerado quando sua entrada é desligada. Usamos uma instrução Reset (R) para zerar este tipo de timer. Ambos contadores continuam contando até o valor 32767 mesmo após alcançarem o preset.

O Off-Delay Timer é usado para desligar uma saída após um período fixo de tempo. Ao ligarmos a sua entrada IN, o timer seta sua saída (bit) em ativo, mas não conta. Ele somente começa a contar ao desligarmos sua entrada. Após a contagem do tempo, ele seta sua saída (bit) para zero (desliga). Este tipo de timer precisa de uma Borda de Descida para começar a contar.

A saída do timer é representada pelo bit (Txxx) que é o seu endereço de memória. Os timers estão disponíveis em 3 resoluções 1, 10 e 100 milissegundos. A tabela abaixo especifica a configuração dessas resoluções:

Timer Type	Resolution in milliseconds (ms)	Maximum Values in seconds (s)	Timer Number
TONR	1ms	32.767	T0, T64
	10ms	327.67	T1 to T4, T65 to T68
	100ms	3276.7	T5 to T31, T69 to T95
TON - TOF	1ms	32.767	T32, T96
	10ms	327.67	T33 to T36, T97 to T100
	100ms	3276.7	T37 to T63, T101 to T255

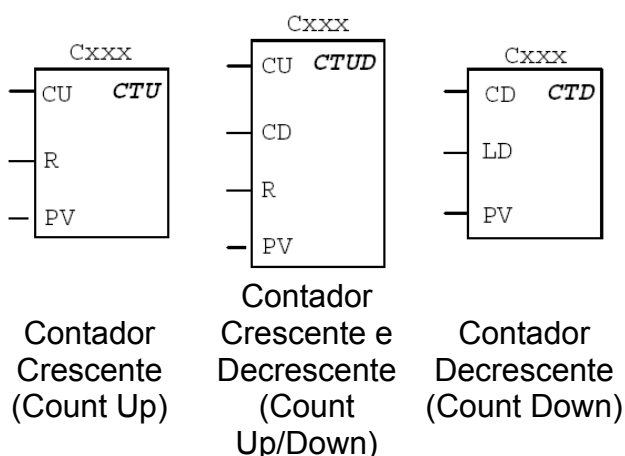
Tabela 6 – Resoluções para Timer.

ATENÇÃO: Você não pode usar o mesmo número de timer para o TON e o TOF (TON T32 e TOF T32, por exemplo)

Inputs / Outputs	Operands	Data Types
IN (LAD)	POWER FLOW	BOOL
PT	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT

Tabela 7 – Operandos de Timer.

CONTADORES (COUNTERS)



O Contador Crescente incrementa em um cada vez que for acionada a sua entrada CU (Count Up). Quando o valor corrente (Cxxx) for maior ou igual ao valor do preset (PV), o bit Cxxx é acionado (liga). O contador crescente é resetado quando acionamos a sua entrada R (Reset).

O Contador Crescente / Decrescente possui duas entradas: CU e CD. Quando acionamos a entrada CU, ele incrementa em um e quando acionamos a entrada CD ele decrementa em um. Quando o valor corrente (Cxxx) for maior ou igual ao valor do preset (PV), o bit Cxxx é acionado (liga). O contador crescente é resetado quando acionamos a sua entrada R (Reset).

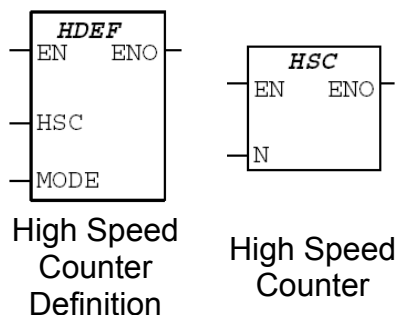
O Contador Crescente Decrementa em um cada vez que for acionada a sua entrada CD (Count Down). Quando o valor corrente (Cxxx) for igual a zero, o bit Cxxx é acionado (liga). O contador decrescente é resetado, ou seja, carrega-se o PV ao Cxxx e sua saída é zerada, quando acionamos a entrada LD (Load).

O número de contadores (Cxxx) vai de 0 até 255.

ATENÇÃO: Você não pode usar o mesmo número de conuter (Cxxx) para o CTU, CTD e CTUD.

Inputs / Outputs	Operands	Data Types
CU, CD	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL
R, LD	I, Q, M, SM, T, C, V, S, L, Power Flow	BOOL
PV	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD,*AC,*LD	INT

Tabela 8 – Operandos de contadores

CONTADORES DE ALTA VELOCIDADE (HIGH SPEED COUNTER)

Um contador HDEF atribui um modo (MODE), de operação referenciado a um contador HSC. A instrução High-Speed Counter (HSC) quando executada, configura e controla a modalidade operacional de um contador de alta velocidade baseada no estado dos bits da memória especial do HSC. O parâmetro N especifica o número do contador de alta velocidade. Apenas uma instrução HDEF deve ser usada por contador. As CPUs 221 e 222 não suportam HSC1 e HSC2.

Inputs / Outputs	Operands	Data Types
HSC	Constant	BYTE
MODE	Constant	BYTE
N	Constant	WORD

Tabela 10 – Operandos de contadores de alta velocidade (HSC)

Estes tipos de contadores contam eventos de altas velocidades que não podem ser controlados pelas taxas de varredura (scans) da CPU e podem ser configurados em mais de 12 modos de operação. A frequência máxima de um contador de alta velocidade depende do tipo de CPU. Os Modos de Operação estão listados nas tabelas abaixo:

HSC0					
MODE	Description	I0.0	I0.1	I0.2	
0	Single Phase up/down counter with internal direction control:	Clock			
1	SM37.3=0, count down SM37.3=1, count up			Reset	
3	Single Phase up/down counter with external direction control:	Clock	Dir		
4	I0.1=0, count down I0.1=1, count up			Reset	
6	Two phase counter with count up and count down inputs	Clock (Up)	Clock (Dn)		
7				Reset	
9	A/B phase quadrature counter: Phase A leads B by 90 degrees for clockwise rotation Phase B leads A by 90 degrees for counterclockwise rotation	Clock Phase A	Clock Phase B		
10				Reset	

Tabela 11 – Modos de operação do contador HSC0

HSC1					
MODE	Description	I0.6	I0.7	I1.0	I1.1
0	Single Phase up/down counter with internal direction control: SM47.3=0, count down SM47.3=1, count up	Clock			
1				Reset	
2					Start
3	Single Phase up/down counter with external direction control: I0.7=0, count down I0.7=1, count up	Clock	Dir		
4				Reset	
5					Start
6	Two phase counter with count up and count down inputs	Clock (Up)	Clock (Dn)		
7				Reset	
8					Start
9	A/B phase quadrature counter: Phase A leads B by 90 degrees for clockwise rotation Phase B leads A by 90 degrees for counterclockwise rotation	Clock Phase A	Clock Phase B		
10				Reset	
11					Start

Tabela 12 – Modos de operação do contador HSC1

HSC2					
MODE	Description	I1.2	I1.3	I1.4	I1.5
0	Single Phase up/down counter with internal direction control: SM57.3=0, count down SM57.3=1, count up	Clock			
1				Reset	
2					Start
3	Single Phase up/down counter with external direction control: I1.3=0, count down I1.3=1, count up	Clock	Dir		
4				Reset	
5					Start
6	Two phase counter with count up and count down inputs	Clock (Up)	Clock (Dn)		
7				Reset	
8					Start
9	A/B phase quadrature counter: Phase A leads B by 90 degrees for clockwise rotation Phase B leads A by 90 degrees for counterclockwise rotation	Clock Phase A	Clock Phase B		
10				Reset	
11					Start

Tabela 13 – Modos de operação do contador HSC2

HSC3					
MODE	Description	I0.1			
0	Single Phase up/down counter with internal direction control: SM137.3=0, count down SM137.3=1, count up	Clock			

Tabela 14 – Modos de operação do contador HSC3

HSC4					
MODE	Description	I0.3	I0.4	I0.5	
0	Single Phase up/down counter with internal direction control: SM147.3=0, count down SM147.3=1, count up	Clock			
1				Reset	
3	Single Phase up/down counter with external direction control: I0.4=0, count down I0.4=1, count up	Clock	Dir		
4				Reset	
6	Two phase counter with count up and count down inputs	Clock (Up)	Clock (Dn)		
7				Reset	
9	A/B phase quadrature counter: Phase A leads B by 90 degrees for clockwise rotation Phase B leads A by 90 degrees for counterclockwise rotation	Clock Phase A	Clock Phase B		
10				Reset	

Tabela 15 – Modos de operação do contador HSC4

HSC5					
MODE	Description	I0.4			
0	Single Phase up/down counter with internal direction control: SM157.3=0, count down SM157.3=1, count up	Clock			

Tabela 16 – Modos de operação do contador HSC5

Na prática, esses contadores servem para contar eventos de alta frequência, como a velocidade de giro de um motor, por exemplo.

A cada interrupção valor-corrente=valor-preset, um novo Preset é carregado e o próximo estado das saídas é setado. Quando a interrupção reset ocorre, o primeiro preset e o primeiro estado da saída é setado e o ciclo se repete. Antes de usar um HSC, você deve selecionar o Counter MODE, que pode ser definido pela instrução HDEF. Todos os contadores suportam interrupções.

Interrupt Point														
Element	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0	1.1	1.2	1.3	1.4	1.5
HSC0	x	x	x											
HSC1							x	x	x	x				
HSC2											x	x	x	x
HSC3		x												
HSC4				x	x	x								
HSC5					x									
Edge Interrupts	x	x	x	x										

Tabela 17 – Entradas atribuídas para os HSC e as interrupções de borda.

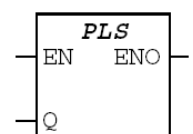
High-Speed Counters	Inputs Used
HSC0	I0.0 - I0.1 - I0.2
HSC1	I0.6 - I0.7 - I1.0 - I1.0
HSC2	I1.2 - I1.3 - I1.4 - I1.5
HSC3	I0.1
HSC4	I0.3 - I0.4 - I0.5
HSC5	I0.4

Tabela 18 – Entradas Dedicadas Para os contadores HSC

HSC0	HSC1	HSC2	HSC4	Description (Used only when HDFE is executed)
SM37.0	SM47.0	SM57.0	SM147.0	Active level control bit for Reset: 0=Reset is active high; 1=Reset is active low
-	SM47.1	SM57.1	-	Active level control bit for Start: 0=Start is active high; 1=Star is active low
SM37.2	SM47.2	SM57.2	SM147.2	Counting rate selection for Quadrature counters: 0=4x counting rate; 1=1x counting rate

Tabela 19 – Nível de ativação para Reset, Start e 1x/4x control bits

INSTRUÇÕES DE SAÍDA DE PULSO (PULSE OUTPUT INSTRUCTIONS)



Pulse Output

A instrução de saída de pulso examina os bits específicos de memória para a configuração das saídas de pulso (Q0.0 e Q0.1). O tipo de pulso é definido pela configuração destas memórias especiais (SM).

As CPUs S7-200 possuem dois tipos de saídas de pulso: PTO (Pulse Train Operation) e PWM (Pulse Width Modulation).

A função PTO oferece uma saída de onda quadrada com 50% duty cycle com o controle pelo usuário do tempo de ciclo e o número de pulsos. A função PWM oferece um duty cycle contínuo ou variável em sua saída com o controle pelo usuário do tempo de ciclo e da largura de pulso.

Cada gerador PTO/PWM possui um BYTE de controle (8 bits), um valor para o tempo de ciclo e largura de pulso (unsigned 16-bit) e um contador de pulsos (unsigned 32-bit). Estes valores estão alocado na área das memória especiais (SM). Uma vez estas memória configuradas com os valores necessários para a configuração das saídas de pulso, estas operações são executadas através da instrução PLS. Esta instrução lê as memórias e programa o PTO/PWM de acordo com seus valores.

Byte Offset From Profile Table Start	Profile Segment Number	Description of Table Entries
0		Number of segments (1 to 255); a value of 0 generates a non-fatal error. No PTO output is generated.
1	#1	Initial cycle time (2 to 65535 units of the time base)
3		Cycle time delta per pulse (signed value) (-32768 to 32767 units of the time base)
5		Pulse count (1 to 4294967295)
9	#2	Initial cycle time (2 to 65535 units of the time base)
11		Cycle time delta per pulse (signed value) (-32768 to 32767 units of the time base)
13		Pulse count (1 to 4294967295)
...

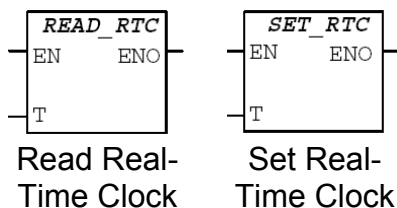
Tabela 20 – Múltiplos segmentos para a operação do PTO.

Antes de inicializar os geradores de pulso, devemos usar a primeira varredura do programa para zerar as duas saídas do CLP.

Q0.0	Q0.0	Status Byte
SM66.4	SM76.4	PTO profile aborted due to delta calculation error. 0 = no error; 1 = aborted
SM66.5	SM76.5	PTO profile aborted due to user command. 0 = no abort; 1 = aborted
SM66.6	SM76.6	PTO pipeline overflow/underflow. 0 = no overflow; 1 = overflow/underflow
SM66.7	SM76.7	PTO idle. 0 = in progress; 1 = PTO idle
Q0.0	Q0.1	Control Byte
SM67.0	SM77.0	PTO/PWM update cycle time value. 0 = no update; 1 = update cycle time.
SM67.1	SM77.1	PWM update pulse width time value. 0 = no update; 1 = update pulse width.
SM67.2	SM77.2	PTO update pulse count value. 0 = no update; 1 = update pulse count.
SM67.3	SM77.3	PTO/PWM time base select. 0 = 1 s/tick; 1 = 1ms/tick
SM67.4	SM77.4	PWM update method. 0 = asynchronous update, 1 = synchronous update
SM67.5	SM77.5	PTO operation. 0 = single segment operation 1 = multiple segment operation
SM67.6	SM77.6	PTO/PWM mode select. 0 = selects PTO; 1 = selects PWM
SM67.7	SM77.7	PTO/PWM enable 0 = disables PTO/PWM; 1 = enables PTO/PWM
Q0.0	Q0.1	Other PTO/PWM Registers
SMW68	SMW78	PTO/PWM cycle time value (range: 2 to 65535)
SMW70	SMW80	PWM pulse width value (range: 0 to 65535)
SMD72	SMD82	PTO pulse count value (range: 1 to 4294967295)
SMB166	SMB176	Number of segment in progress (used only in multiple segment PTO operation)
SMW168	SMW178	Starting location of profile table, expressed as a byte offset from V0 (used only in multiple segment PTO operation)

Tabela 21 – PTO/PWM control registers.

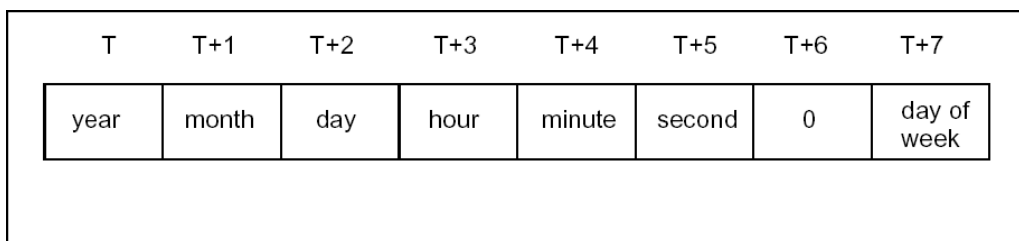
INSTRUÇÕES DE CLOCK (RELÓGIO DE TEMPO REAL)



Read Real-Time Clock faz a leitura da hora e da data atual e carrega num buffer de 8 Bytes (começando em um endereço T).

Set Real-Time Clock faz a escrita da hora e da data atual e carrega num buffer de 8 Bytes (começando em um endereço T).

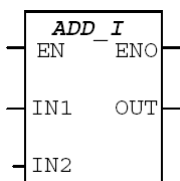
Inputs / Outputs	Operands	Data Types
T	VB, IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD	BYTE



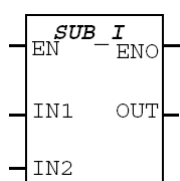
Formatos do Time Buffer

Year/Month	yymm	yy - 0 to 99	mm - 1 to 12
Day/Hour	ddhh	dd - 1 to 31	hh - 0 to 23
Minute/Second	mmss	mm - 0 to 59	ss - 0 to 59
Day of week	d	d - 0 to 7	1 = Sunday
			0 = disables day of week (remains 0)

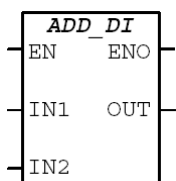
Data Formats

INSTRUÇÕES MATEMÁTICAS

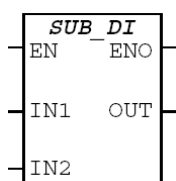
Add Integer



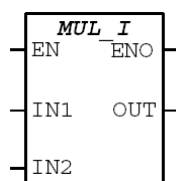
Subtract Integer



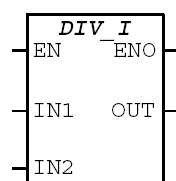
Add Double Integer



Subtract Double Integer



Multiply Integer



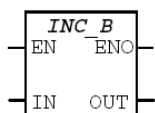
Divide Integer

IN1 (instrução matemática) IN2 = OUT

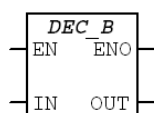
Integer = 16 bits

Double Integer = 32 bits

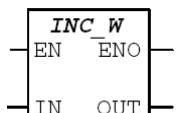
Inputs / Outputs	Operands	Data Types
Add/Subtract Integer		
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	INT
Add/Subtract Double Integer		
IN1, IN2	VD, ID, QD, MD, SMD, SD, LD, AC, HC, Constant, *VD, *AC, *LD	DINT
OUT	VD, ID, QD, MD, SM, SD, LD, AC, *VD, *AC, *LD	DINT
Multiply/Divide Integer		
IN1, IN2	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	INT

INSTRUÇÕES INCREMENTA / DECREMENTA

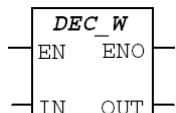
Incrementa BYTE



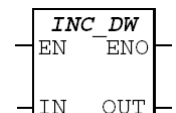
Decrementa BYTE



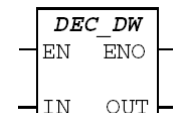
Incrementa WORD



Decrementa WORD



Incrementa DOUBLE WORD

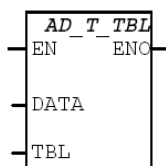


Decrementa DOUBLE WORD

A instrução increment/decrement serve para respectivamente incrementar e decrementar em 1 o valor da entrada IN e coloca o resultado em OUT.

Inputs / Outputs	Operands	Data Types
BYTE		
IN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
WORD		
IN	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	INT
DOUBLE WORD		
IN	VD, ID, QD, MD, SMD, SD, LD, AC, HC, Constant, *VD, *AC, *LD	DINT
OUT	VD, ID, QD, MD, SM, SD, LD, AC, *VD, *AC, *LD	DINT

INSTRUÇÕES DE TABELA (TABLE INSTRUCTIONS)

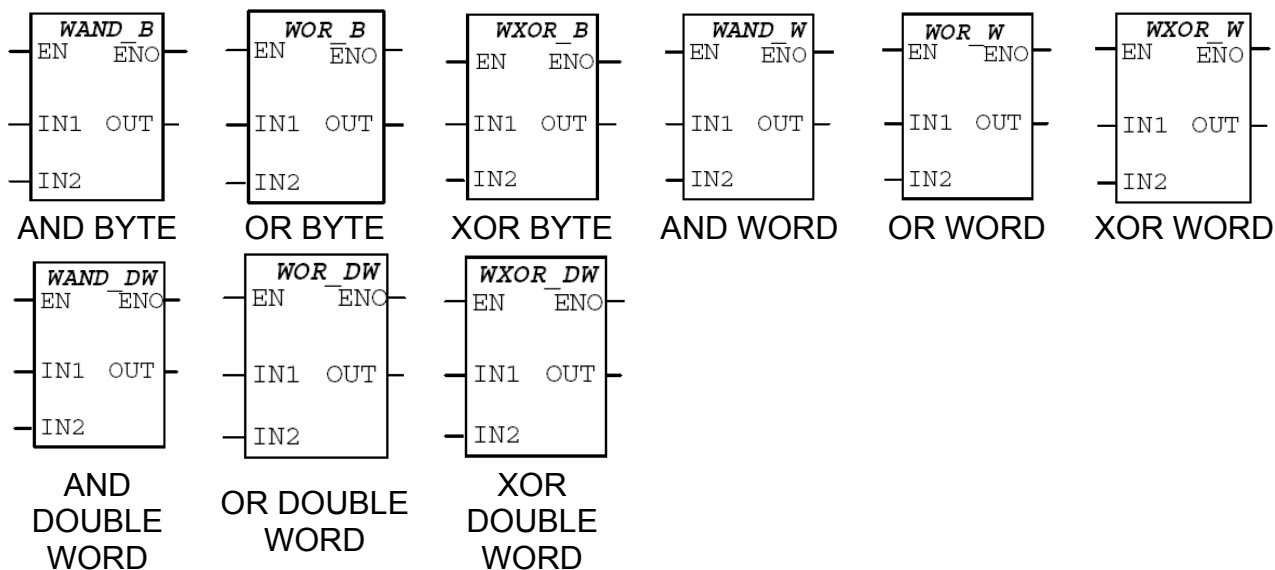


Add To Table

Essa instrução adiciona um dado (DATA) para uma tabela. O primeiro valor da tabela é o valor máximo de comprimento da tabela. O segundo valor é o Entry Count (EC), que especifica o número de entradas na tabela. Novos dados são adicionados à tabela após a última entrada. A cada novo dado adicionado à tabela, o EC é incrementado. A tabela suporta mais de 10 entradas de dados.

Inputs / Outputs	Operands	Data Types
	BYTE	
IN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
	WORD	
IN	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	INT
	DOUBLE WORD	
IN	VD, ID, QD, MD, SMD, SD, LD, AC, HC, Constant, *VD, *AC, *LD	DINT
OUT	VD, ID, QD, MD, SM, SD, LD, AC, *VD, *AC, *LD	DINT

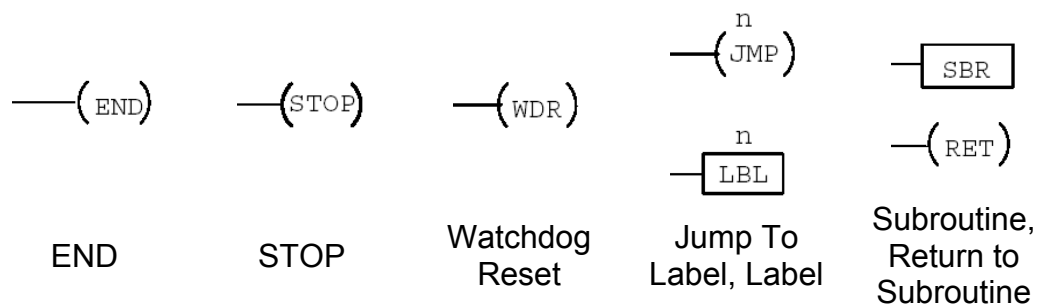
INSTRUÇÕES DE OPERAÇÕES LÓGICAS (LOGICAL OPERATIONS)



As instruções AND, OR e XOR analisam as entradas IN1 e IN2, processam a lógica correspondente e colocam na saída OUT sem alterar o resultado das entradas. Elas afetam a memória especial SM 1.0.

Inputs / Outputs	Operands	Data Types
BYTE		
IN	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD	BYTE
OUT	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
WORD		
IN	VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *AC, *LD	WORD
OUT	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, *VD, *AC, *LD	WORD
DOUBLE WORD		
IN	VD, ID, QD, MD, SMD, SD, LD, AC, HC, Constant, *VD, *AC, *LD	DWORD
OUT	VD, ID, QD, MD, SM, SD, LD, AC, *VD, *AC, *LD	DWORD

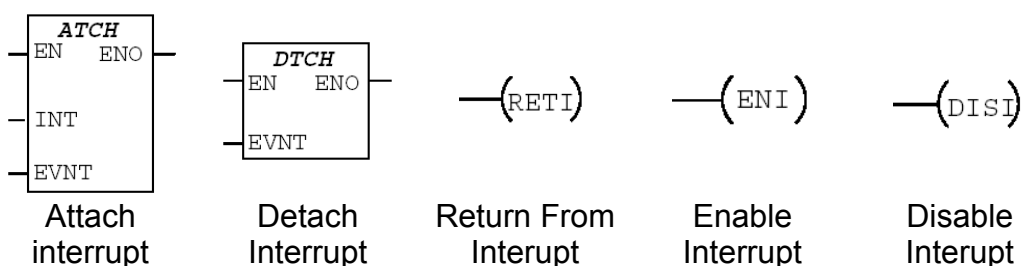
INSTRUÇÕES DE CONTROLE (CONTROL INSTRUCTIONS)



A instrução END finaliza um programa que esteja rodando na cpu. STOP, coloca a CPU em STOP MODE. WDR é o watchdog timer, que reset a CPU após um certo tempo, caso o programa trave: Ele deve ser zerado de tempos em tempos para não resetar a máquina, caso ela esteja funcionando corretamente.

As instruções JMP e SBR, são instruções de salto, ou seja, de desvio do programa principal. JMP é usado para “pular” para uma outra parte do programa, sem retornar ao ponto de pulo. SBR serve para instruções de subrotinas, usadas para auxiliar o programa principal, voltando ao ponto de partida.

INTERRUPÇÕES



ATCH associa um evento de interrupção (EVNT) à um número de uma rotina de interrupção e habilita o evento de interrupção.

DTCH desassocia um evento de interrupção de todas as rotinas de interrupção e desabilita o evento de interrupção.

Inputs / Outputs	Operands	Data Types
INT	Constant (CPU 222: 0-12, 19-23, 27-33; CPU 224: 0-23, 27-33)	BYTE
EVNT	Constant (CPU 222: 0-12, 19-23, 27-33; CPU 224: 0-23, 27-33)	BYTE

Antes que uma rotina da interrupção possa ser invocada, uma associação deve ser estabelecida entre o evento da interrupção e o segmento de programa que você quer executar quando o evento ocorre. Use a instrução da interrupção do anexo (ATCH) para associar um evento da interrupção (especificado pelo número do evento da interrupção) e o segmento de programa (especificado por um número da rotina de interrupção). Você pode unir eventos múltiplos da interrupção a uma rotina da interrupção, mas um evento não pode simultaneamente ser unido às múltiplas rotinas da interrupção. Quando um evento ocorrer com as interrupções permitidas, apenas a última rotina da interrupção unida a este evento será executada.

Quando você une um evento da interrupção a uma rotina da interrupção, essa interrupção está permitida automaticamente. Se você incapacitar todas as interrupções usando global disable interrupt, cada ocorrência do evento da interrupção estão enfileiradas até que as interrupções re-estejam permitidas, usando o global enable interruption.

ENI, habilita todos os processos de interrupção anexados aos seus respectivos eventos. DISI, desabilita todas as interrupções anexadas aos seus respectivos eventos.

A instrução RETI serve para retornar de uma interrupção, baseada na condição da lógica precedente.

As interrupções de I/O incluem borda de subida / descida, high speed counters e PTO. A CPU pode gerar uma interrupção em uma borda de subida / descida de uma entrada ou de eventos que podem ser capturados destes pontos de entrada. Esta aplicação de borda de subida / descida pode ser usada em para uma atenção imediata quando um evento ocorre. Os pontos de I/O suportados são: I0.0 até I0.3

Uma interrupção pode ser gerada pelos TIMER T32 e T96

ATENÇÃO: Você não pode usar as instruções DISI, ENI, HDEF, LSCR e END em uma rotina da interrupção.

Event Number	Interrupt Description	CPU 221	CPU 222	CPU 224
0	Rising edge, I0.0	Y	Y	Y
1	Falling edge, I0.0	Y	Y	Y
2	Rising edge, I0.1	Y	Y	Y
3	Falling edge, I0.1	Y	Y	Y
4	Rising edge, I0.2	Y	Y	Y
5	Falling edge, I0.2	Y	Y	Y
6	Rising edge, I0.3	Y	Y	Y
7	Falling edge, I0.3	Y	Y	Y
8	Port 0: Receive character	Y	Y	Y
9	Port 0: Transmit complete	Y	Y	Y
10	Timed interrupt 0, SMB34	Y	Y	Y
11	Timed interrupt 1, SMB35	Y	Y	Y
12	HSC0 CV=PV (current value = preset value)	Y	Y	Y
13	HSC1 CV=PV (current value = preset value)			Y
14	HSC1 direction changed			Y
15	HSC1 external reset			Y
16	HSC2 CV=PV (current value = preset value)			Y
17	HSC2 direction changed			Y
18	HSC2 external reset			Y
19	PLS0 pulse count complete interrupt	Y	Y	Y
20	PLS1 pulse count complete interrupt	Y	Y	Y
21	Timer T32 CT=PT interrupt	Y	Y	Y
22	Timer T96 CT=PT interrupt	Y	Y	Y
23	Port 0: Receive message complete	Y	Y	Y
24	Port 1: Receive message complete			
25	Port 1: Receive character			
26	Port 1: Transmit complete			
27	HSC0 direction changed	Y	Y	Y
28	HSC0 external reset	Y	Y	Y
29	HSC4 CV=PV (current value = preset value)	Y	Y	Y
30	HSC4 direction changed	Y	Y	Y
31	HSC4 external reset	Y	Y	
32	HSC3 CV=PV (current value = preset value)	Y	Y	Y
33	HSC5 CV=PV (current value = preset value)	Y	Y	Y

Tabela 22 – Diferentes tipos de interrupção (Interrupt Events)

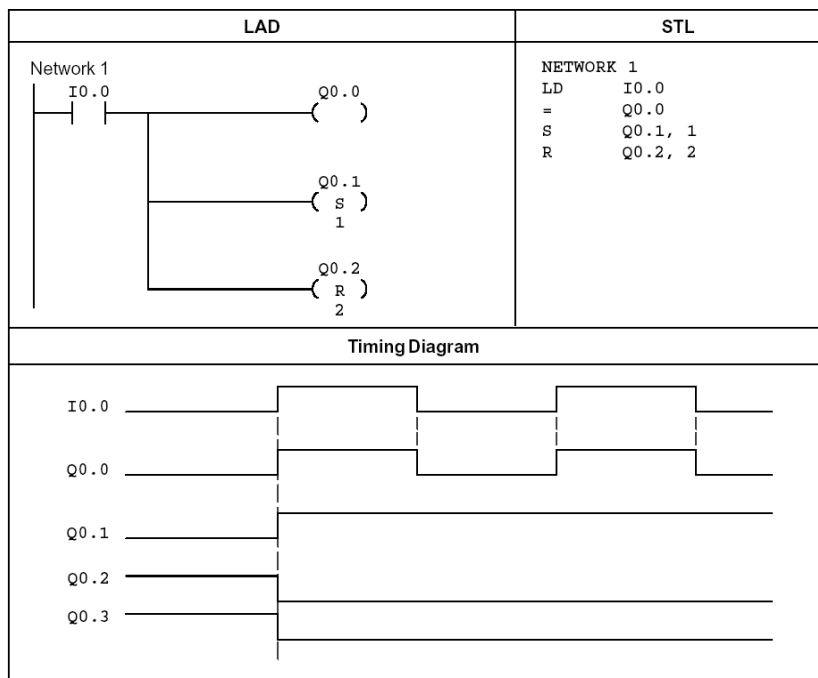
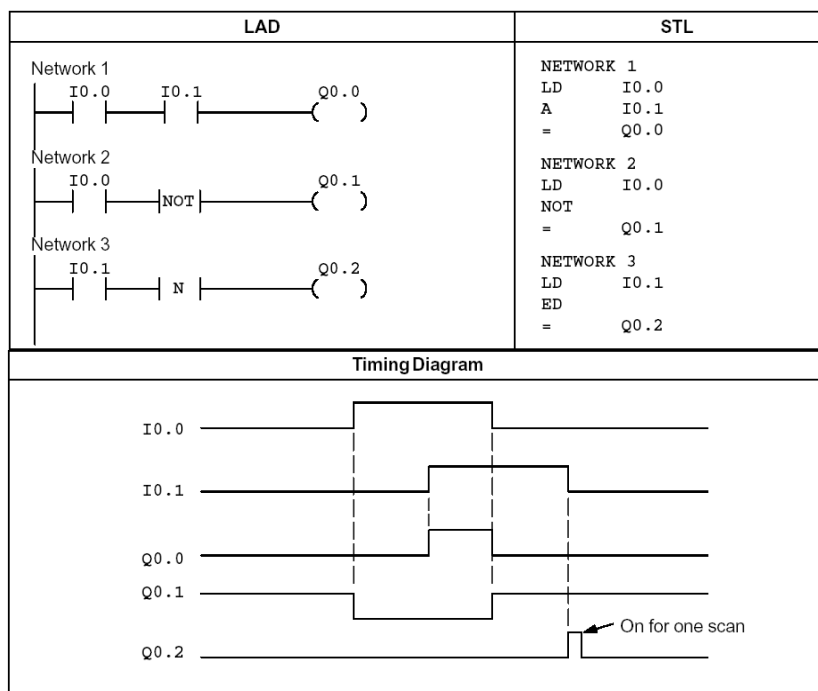
Event Number	Interrupt Description	Priority Group	Priority In Group
8	Port 0: Receive character	Communications (highest)	0
9	Port 0: Transmit complete		0
23	Port 0: Receive message complete		0
24	Port 1: Receive message complete		1
25	Port 1: Receive character		1
26	Port 1: Transmit complete		1
19	PTO 0 complete interrupt	Discrete (middle)	0
20	PTO 1 complete interrupt		1
0	Rising edge, I0.0		2
2	Rising edge, I0.1		3
4	Rising edge, I0.2		4
6	Rising edge, I0.3		5
1	Falling edge, I0.0		6
3	Falling edge, I0.1		7
5	Falling edge, I0.2		8
7	Falling edge, I0.3		9
12	HSC0 CV=PV (current value = preset value)		10
27	HSC0 direction changed		11
28	HSC0 external reset		12
13	HSC1 CV=PV (current value = preset value)		13
14	HSC1 direction input changed		14
15	HSC1 external reset		15
16	HSC2 CV=PV		16
17	HSC2 direction changed		17
18	HSC2 external reset		18
32	HSC3 CV=PV (current value = preset value)		19
29	HSC4 CV=PV (current value = preset value)		20
30	HSC4 direction changed		21
31	HSC4 external reset		22
33	HSC5 CV=PV (current value = preset value)		23
10	Timed interrupt 0	Timed (lowest)	0
11	Timed interrupt 1		1
21	Timer T32 CT=PT interrupt		2
22	Timer T96 CT=PT interrupt		3

Tabela 23 – Eventos de Interrupção em ordem de prioridade.

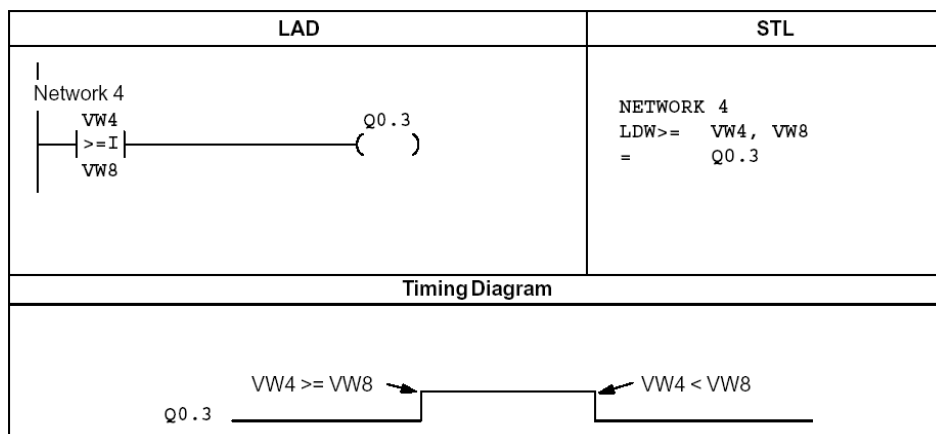
6.5 – EXEMPLOS DAS INSTRUÇÕES LADDER

Estão listados a seguir, exemplos de todas as instruções do CLP SIEMENS SIMATIC S7-200 com os seus respectivos diagramas de tempo (timing diagram).

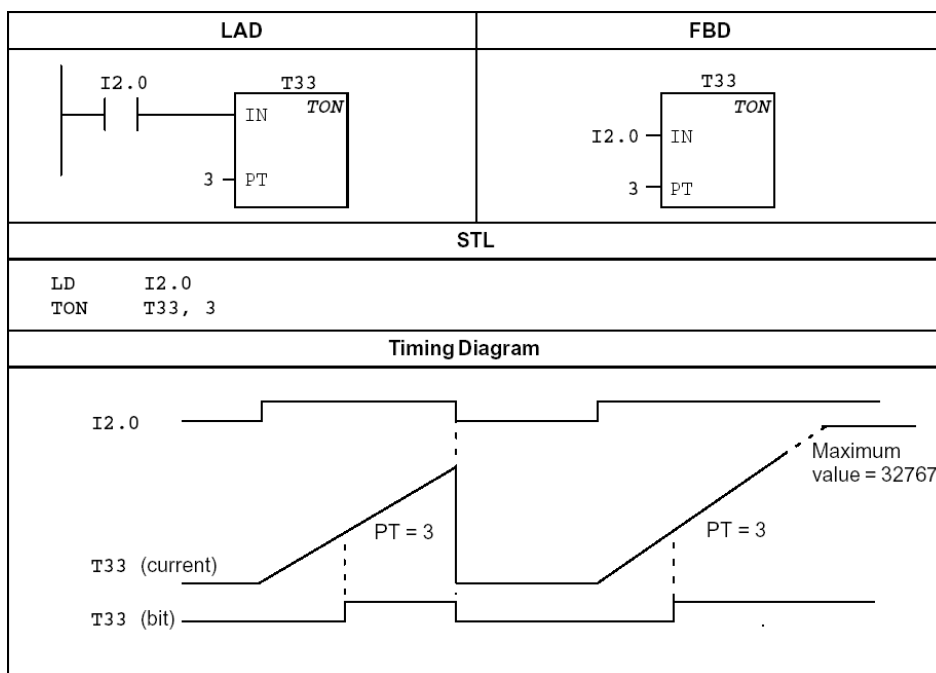
INTRUÇÕES DE LÓGICA BIT (BIT LOGIC INSTRUCTIONS)



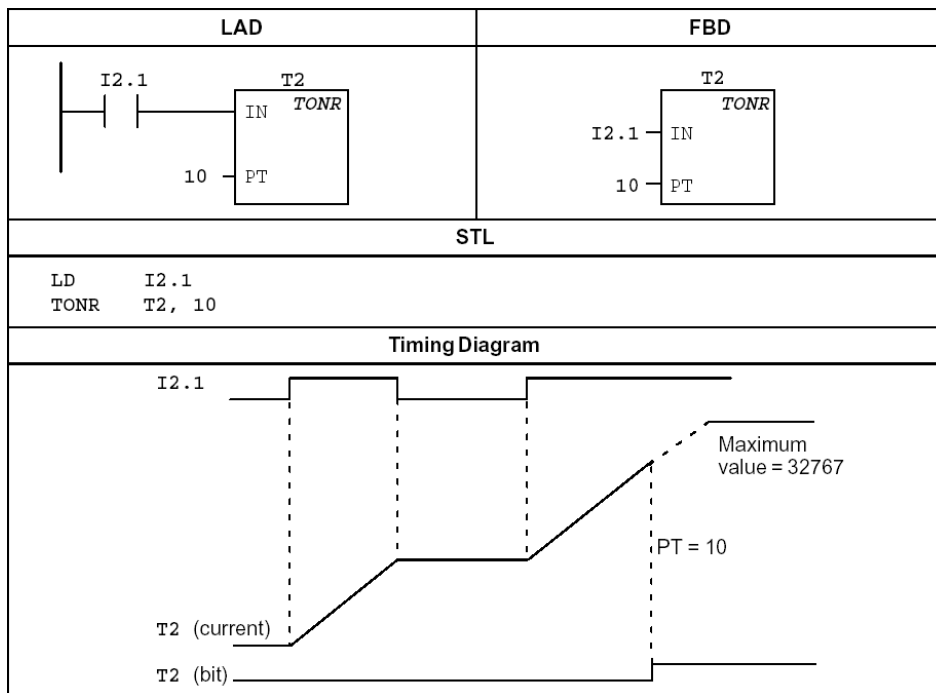
INSTRUÇÕES DE COMPARAÇÃO (COMPARE INSTRUCTIONS)



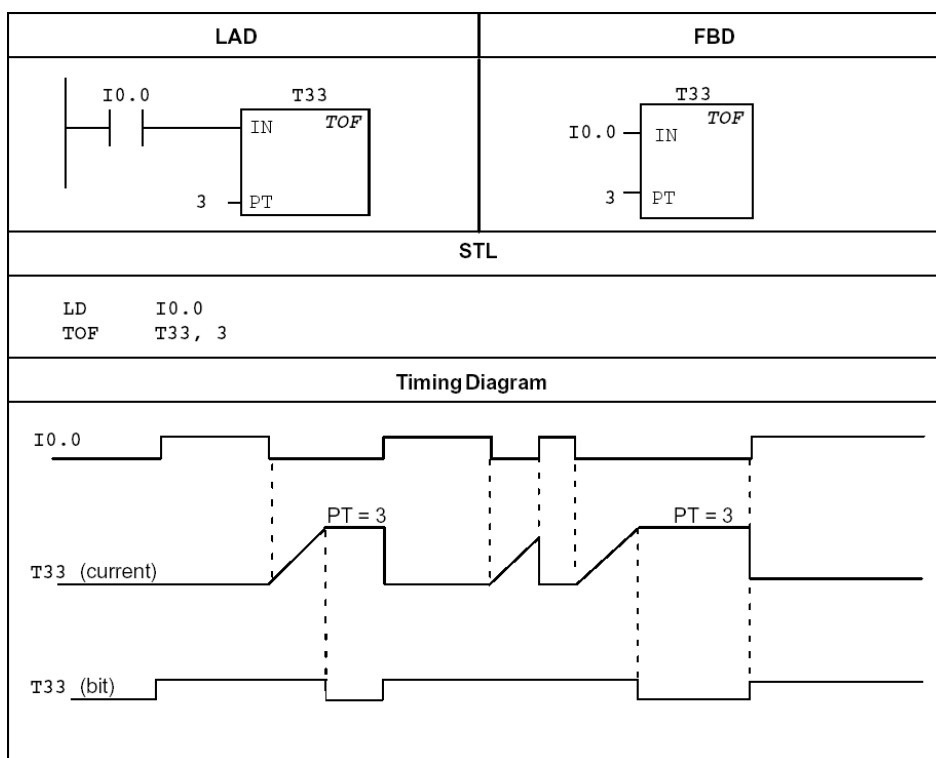
TIMER



On Delay Timer

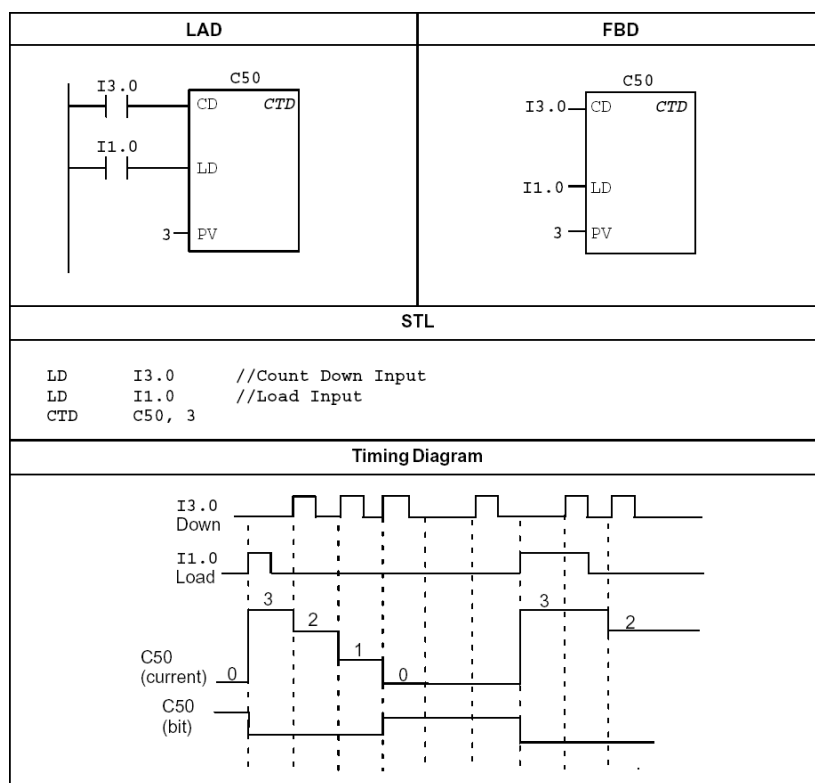


Retentive On Delay Timer

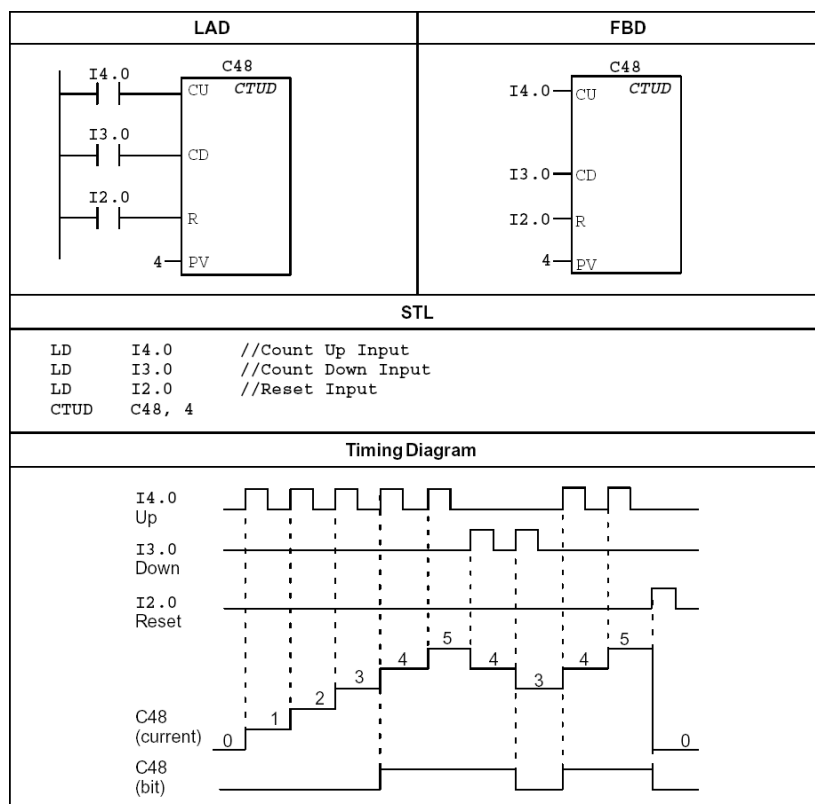


Off Delay Timer

CONTADORES (COUNTERS)



CTD Counter



CTDU Counter

CONTADORES DE ALTA VELOCIDADE (HIGH SPEED COUNTER)

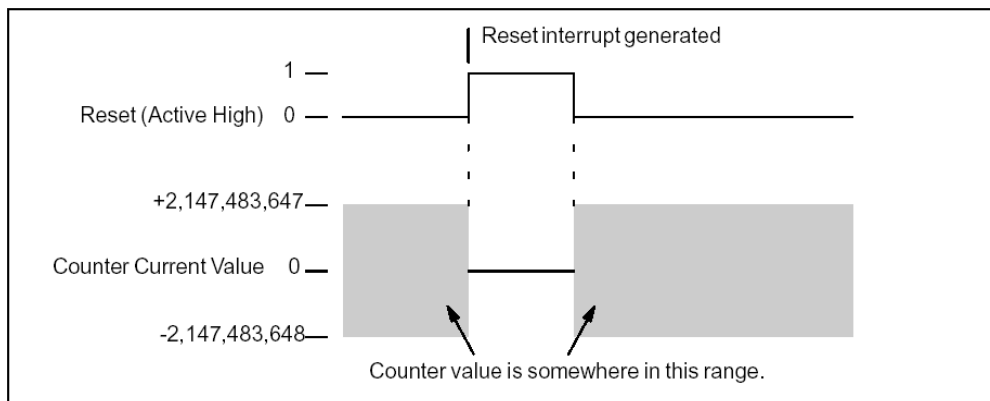


Diagrama com Reset sem Start

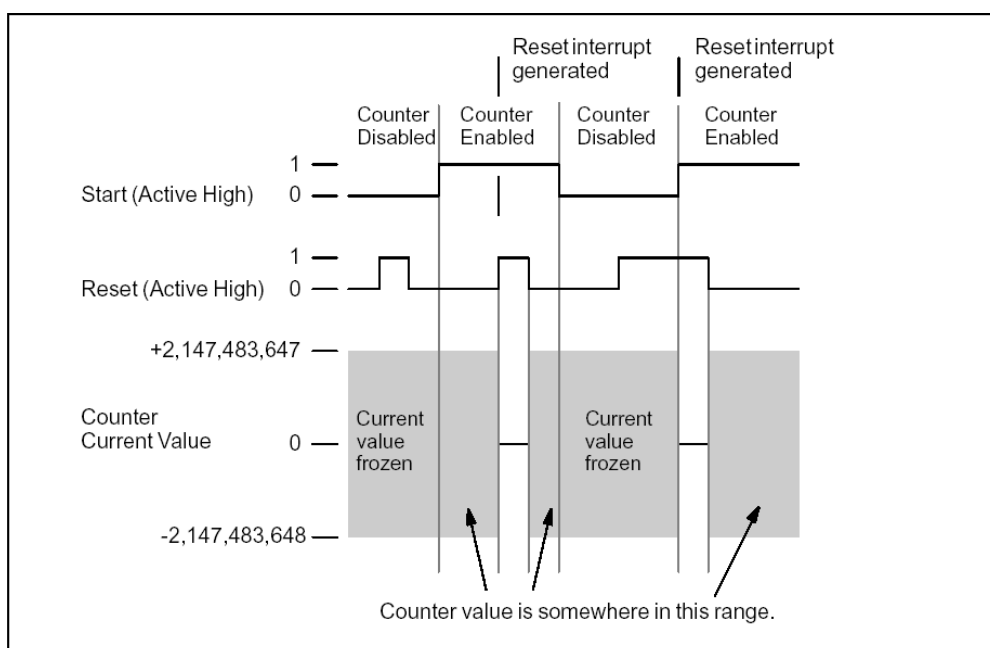
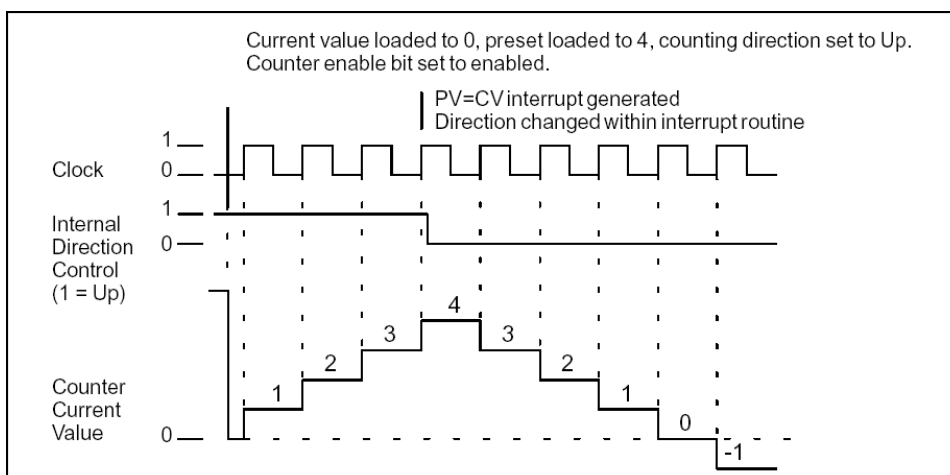
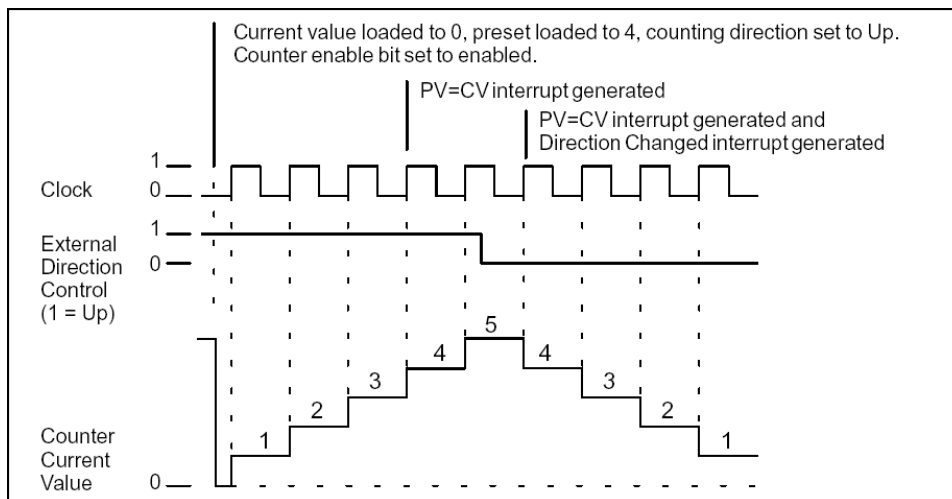


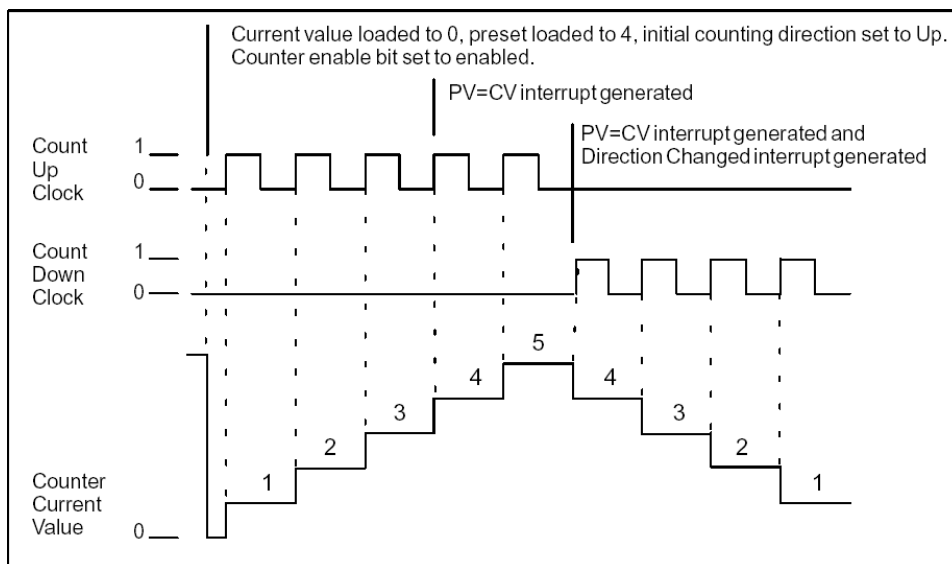
Diagrama com Reset e Start



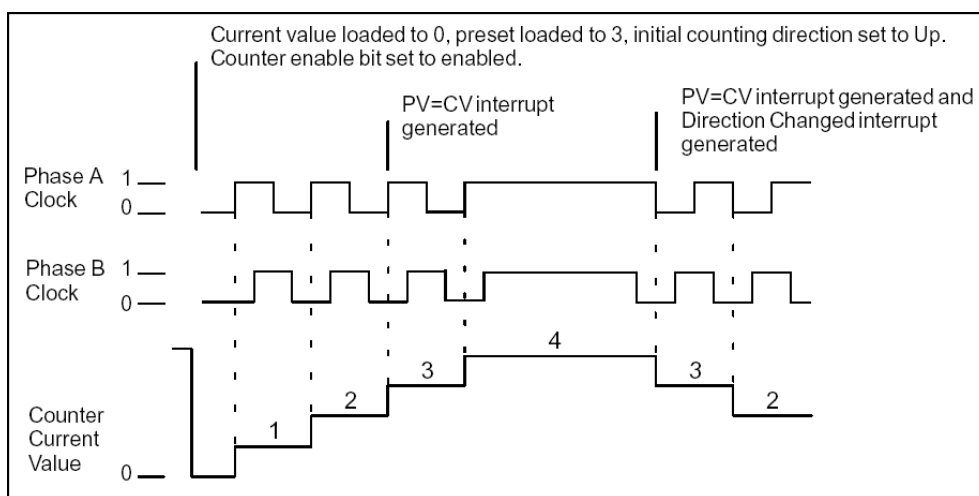
Operação exemplo dos modos 0, 1 ou 2



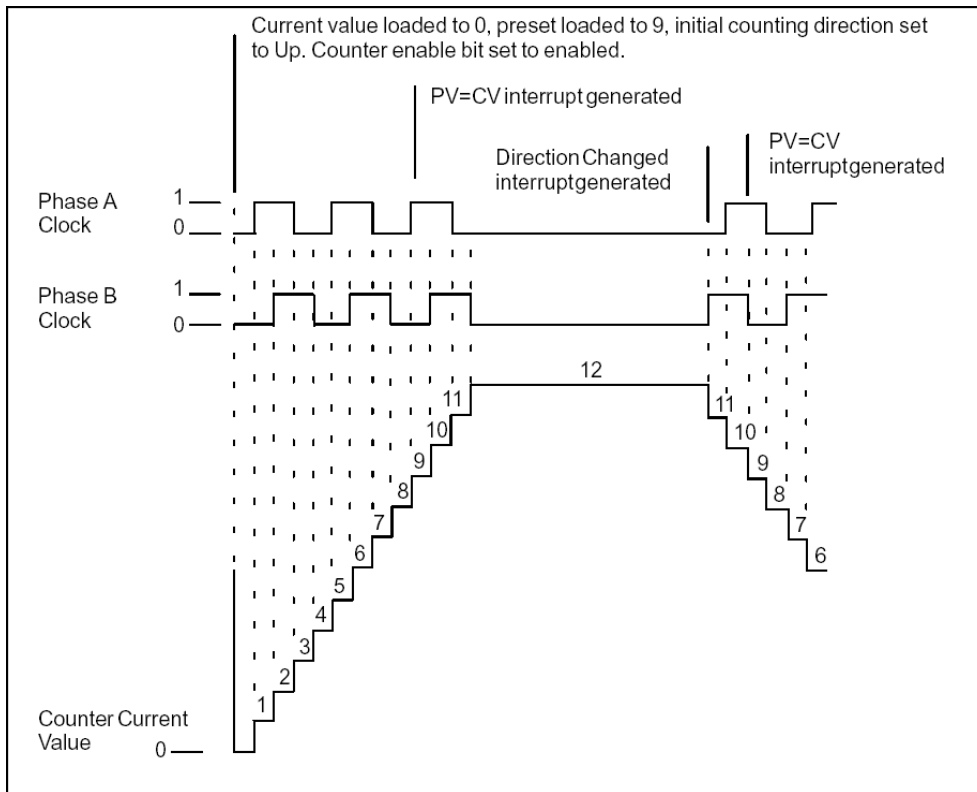
Operação exemplo dos modos 3, 4 ou 5



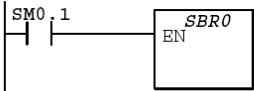
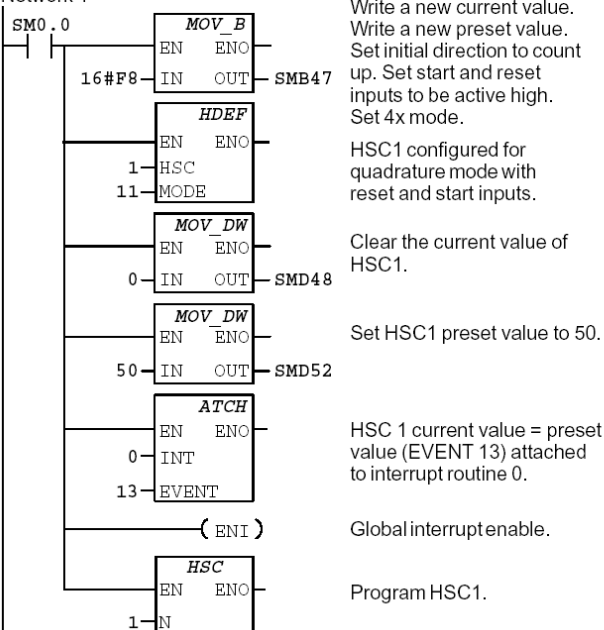
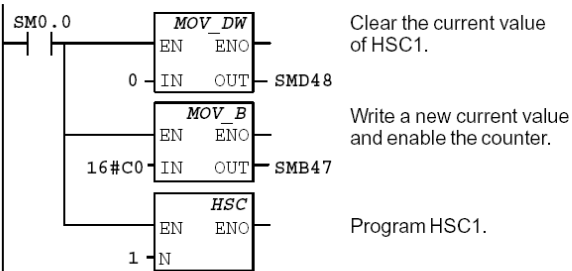
Operação exemplo dos modos 6, 7 ou 8



Operação exemplo dos modos 9, 10 ou 11(Quadrature 1x)

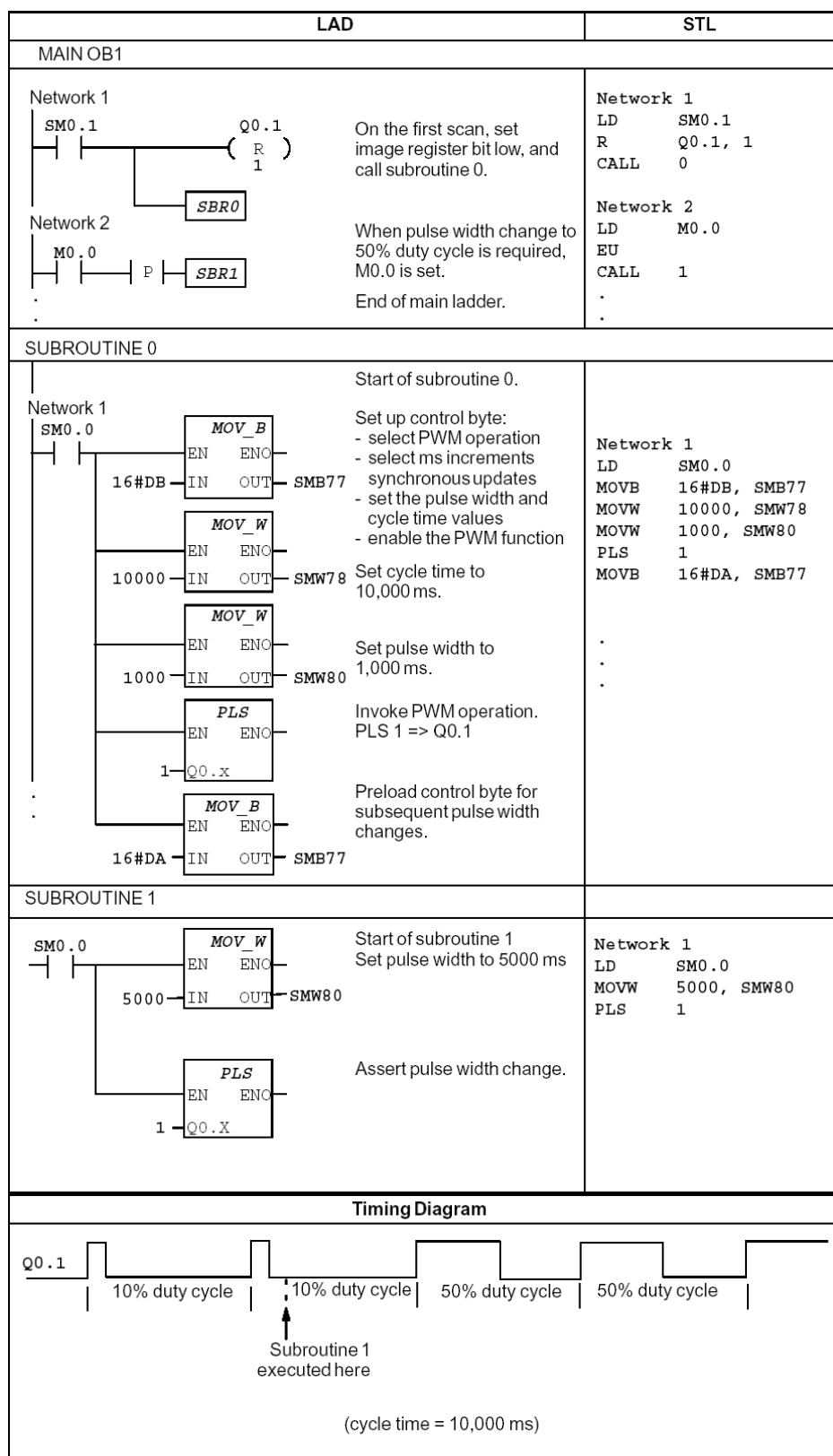


Operação exemplo dos modos 9, 10 ou 11(Quadrature 4x)

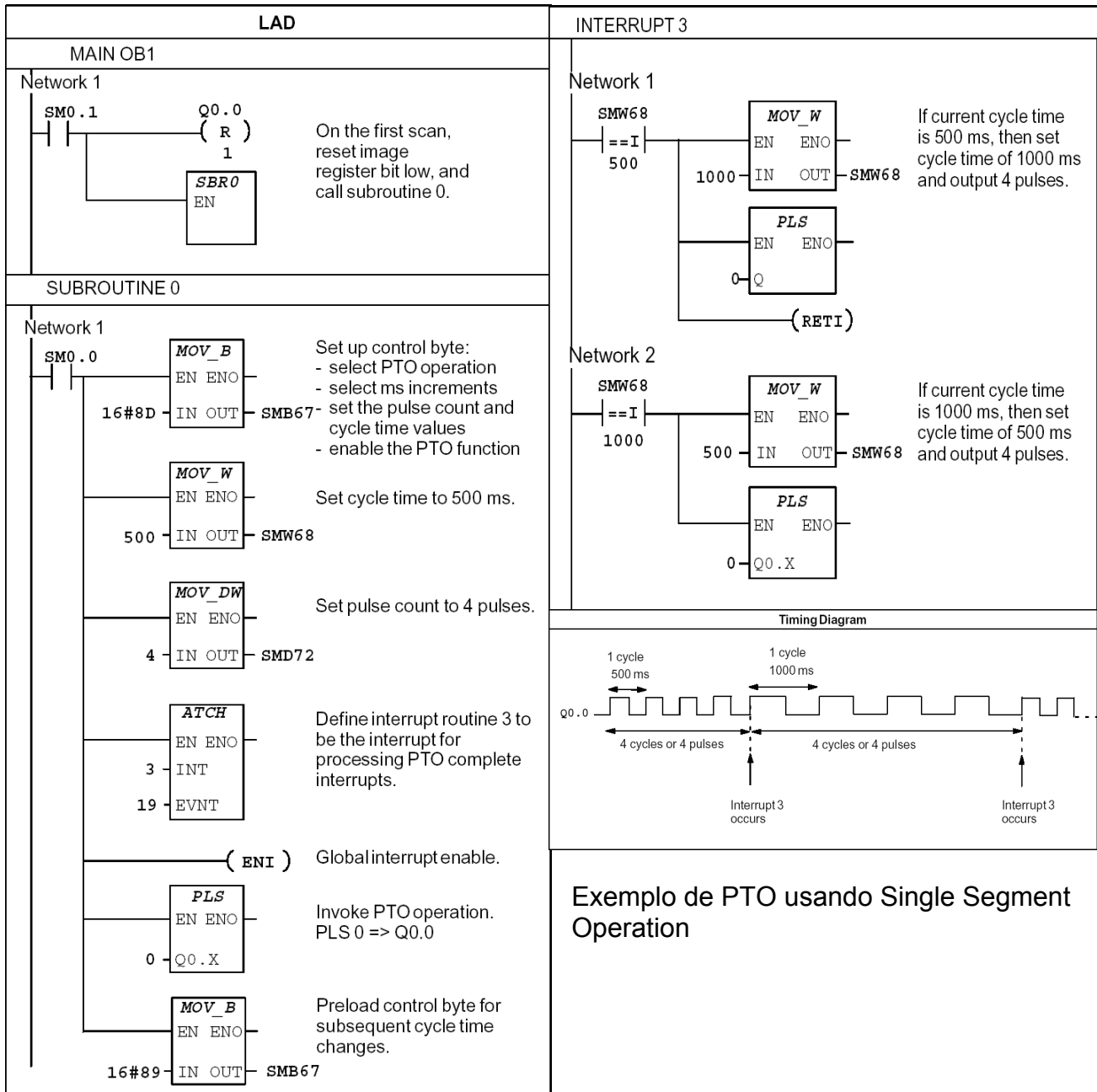
LAD	STL
MAIN OB1	
<p>Network 1</p>  <p>On the first scan, call subroutine 0.</p> <p>End of main program.</p>	<p>Network 1</p> <pre>LD SM0.1 CALL 0</pre>
SUBROUTINE 0	
<p>Network 1</p>  <p>Enable the counter. Write a new current value. Write a new preset value. Set initial direction to count up. Set start and reset inputs to be active high. Set 4x mode.</p> <p>HSC1 configured for quadrature mode with reset and start inputs.</p> <p>Clear the current value of HSC1.</p> <p>Set HSC1 preset value to 50.</p> <p>HSC 1 current value = preset value (EVENT 13) attached to interrupt routine 0.</p> <p>Global interruptenable.</p> <p>Program HSC1.</p>	<p>Network 1</p> <pre>LD SM0.0 MOVB 16#F8, SMB47 HDEF 1, 11 MOVD 0, SMD48 MOVD 50, SMD52 ATCH 0, 13 ENI HSC 1</pre>
INTERRUPT 0	
<p>Network 1</p>  <p>Clear the current value of HSC1.</p> <p>Write a new current value and enable the counter.</p> <p>Program HSC1.</p>	<p>Network 1</p> <pre>LD SM 0.0 MOVD 0, SMD48 MOVB 16#C0, SMB47 HSC 1</pre>

Exemplo de inicialização do HSC1

INSTRUÇÕES DE SAÍDA DE PULSO (PULSE OUTPUT INSTRUCTIONS)

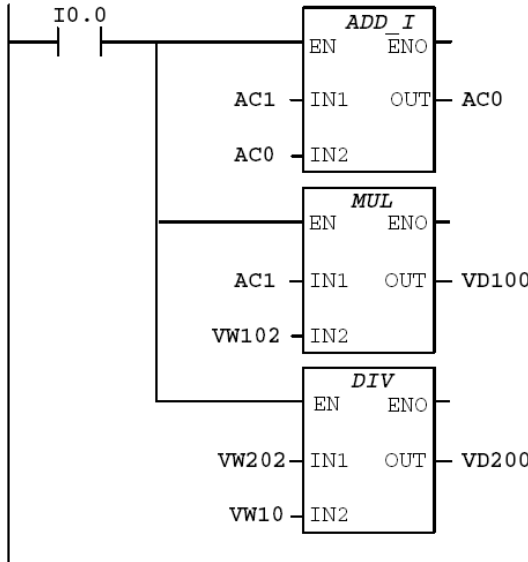
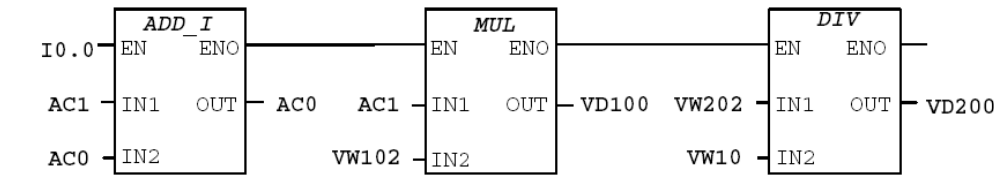


Exemplo de uma saída de alta velocidade usando PWM.



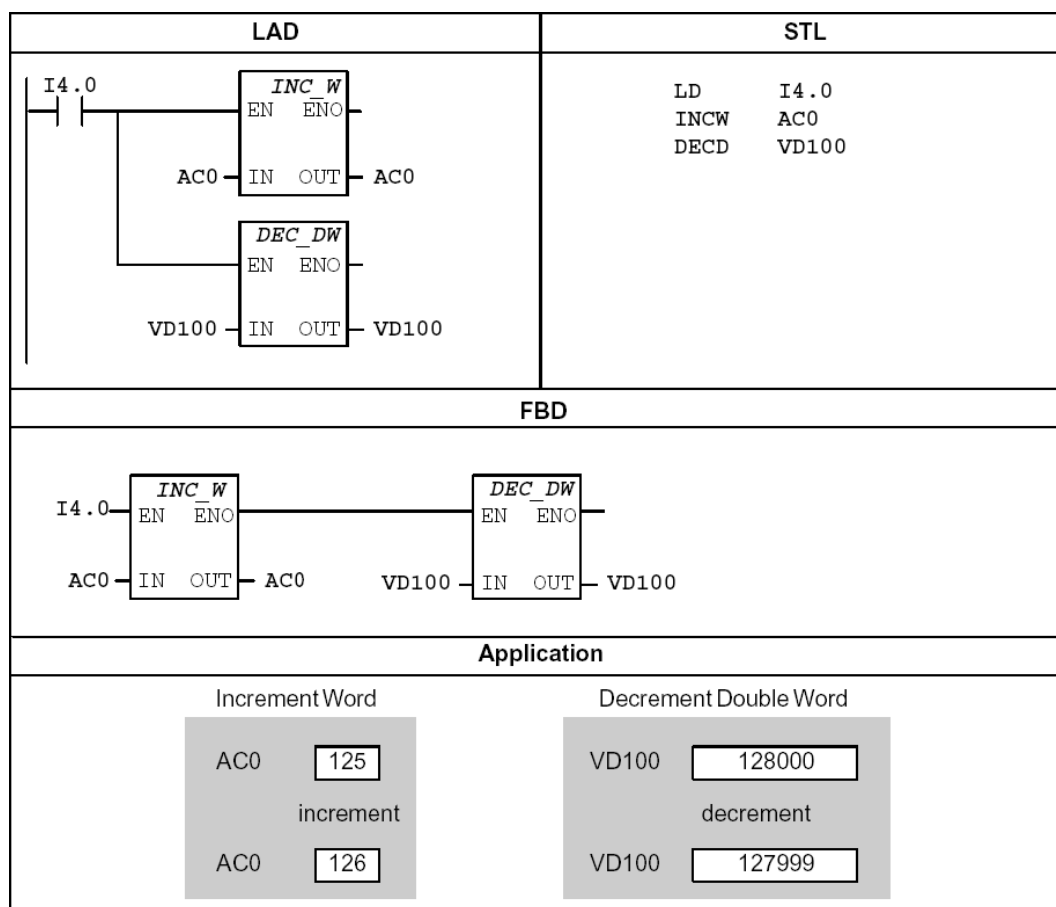
Exemplo de PTO usando Single Segment Operation

INSTRUÇÕES MATEMÁTICAS

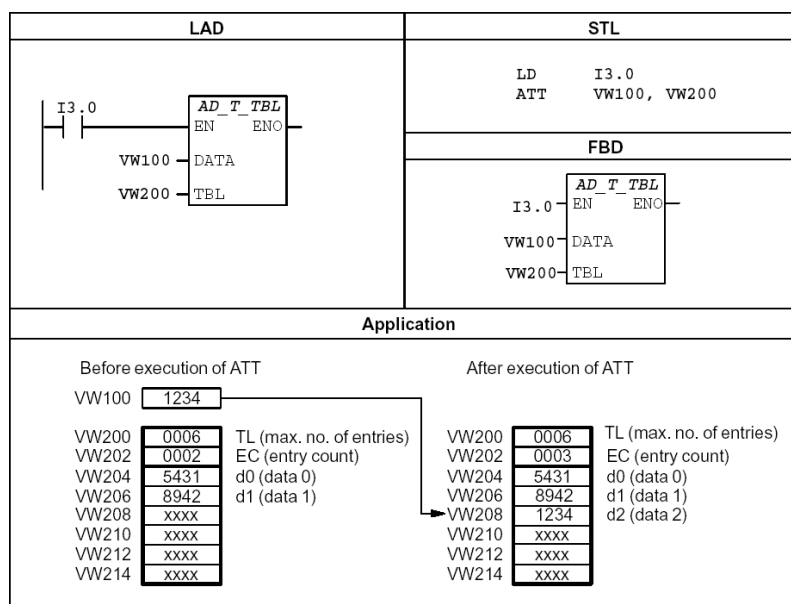
LAD	STL																																				
<p>Network 1</p> 	<pre>NETWORK 1 LD I0.0 +I AC1, AC0 MUL AC1, VD100 DIV VW10, VD200</pre>																																				
FBD																																					
<p>Network 1</p> 																																					
Application																																					
<p>Add</p> <table><tr><td>AC1</td><td>4000</td></tr><tr><td colspan="2">plus</td></tr><tr><td>AC0</td><td>6000</td></tr><tr><td colspan="2">equals</td></tr><tr><td>AC0</td><td>10000</td></tr></table>	AC1	4000	plus		AC0	6000	equals		AC0	10000	<p>Multiply</p> <table><tr><td>AC1</td><td>4000</td></tr><tr><td colspan="2">multiplied by</td></tr><tr><td>VD100</td><td>200</td></tr><tr><td colspan="2">equals</td></tr><tr><td>VD100</td><td>800000</td></tr></table>	AC1	4000	multiplied by		VD100	200	equals		VD100	800000	<p>Divide</p> <table><tr><td>VD200</td><td>4000</td></tr><tr><td colspan="2">divided by</td></tr><tr><td>VW10</td><td>41</td></tr><tr><td colspan="2">equals</td></tr><tr><td>VD200</td><td>23</td><td>97</td></tr><tr><td colspan="2">rem. quot.</td></tr><tr><td>VW200</td><td>VW202</td></tr></table>	VD200	4000	divided by		VW10	41	equals		VD200	23	97	rem. quot.		VW200	VW202
AC1	4000																																				
plus																																					
AC0	6000																																				
equals																																					
AC0	10000																																				
AC1	4000																																				
multiplied by																																					
VD100	200																																				
equals																																					
VD100	800000																																				
VD200	4000																																				
divided by																																					
VW10	41																																				
equals																																					
VD200	23	97																																			
rem. quot.																																					
VW200	VW202																																				
<p>Note: VD100 contains VW100 and VW102. VD200 contains VW200 and VW202.</p>																																					

Adição, Multiplicação e Divisão

INSTRUÇÕES INCRMENTA / DECREMENTA

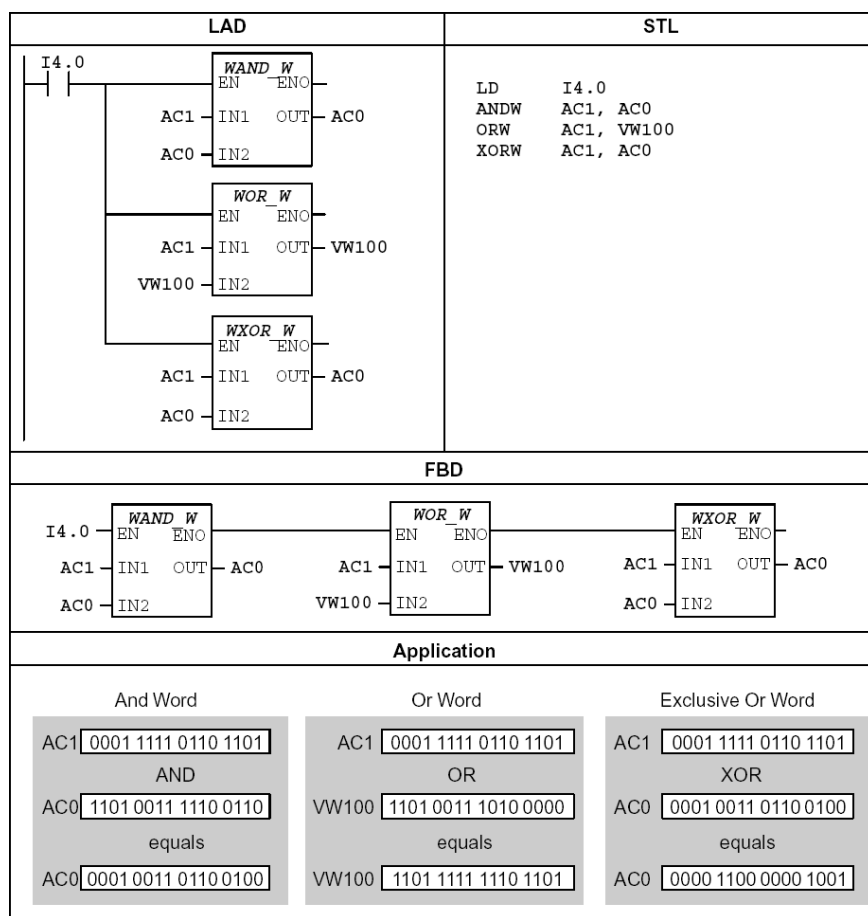


INSTRUÇÕES DE TABELA

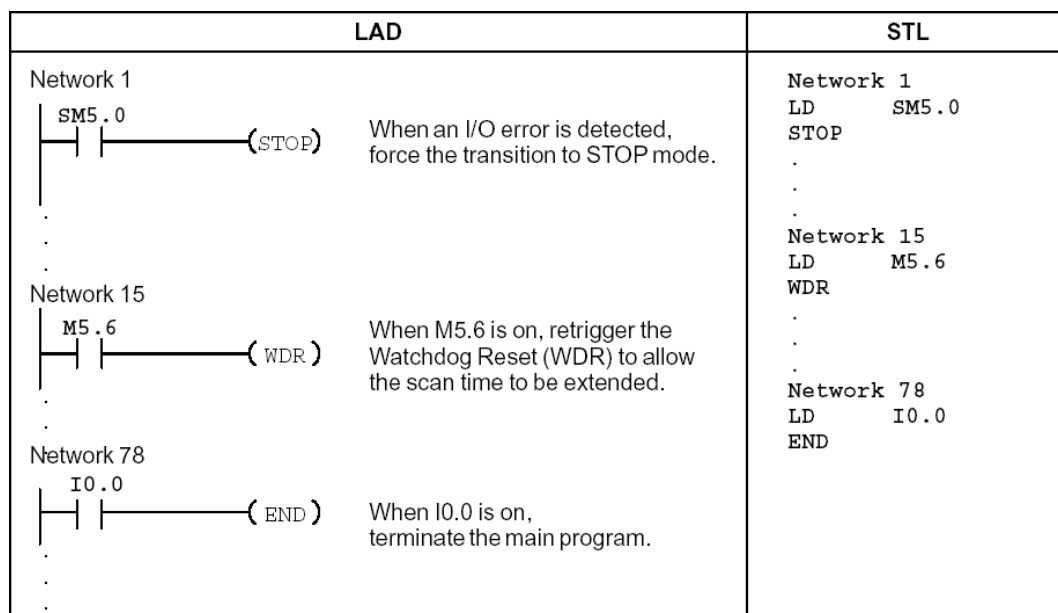


Add To Table

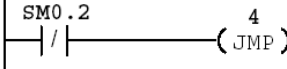
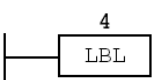
INSTRUÇÕES LÓGICAS



INSTRUÇÕES DE CONTROLE



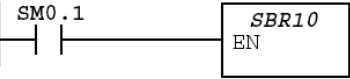
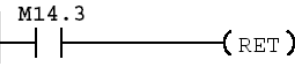
Stop, WDR e End

LAD	STL
<p>Network 14</p>  <p>·</p> <p>·</p> <p>·</p> <p>Network 33</p> 	<p>Network</p> <p>LDN SM0.2</p> <p>JMP 4</p> <p>·</p> <p>·</p> <p>·</p> <p>Network</p> <p>LBL 4</p>

If the retentive data has not been lost, jump to LBL 4.

You can use the JMP to LBL instruction in the main program, in subroutines, or in interrupt routines. The JMP and its corresponding LBL must always be located within the same segment of code (either the main program, a subroutine, or an interrupt routine).

JUMP TO LABEL, LABEL

LAD	STL
MAIN	
<p>Network 1</p>  <p>·</p>	<p>Network 1</p> <p>LD SM0.1</p> <p>CALL 10</p> <p>·</p>
SUBROUTINE 10	
<p>·</p> <p>·</p> <p>·</p> <p>Network 6</p>  <p>·</p> <p>·</p> <p>·</p>	<p>·</p> <p>·</p> <p>·</p> <p>Network 6</p> <p>LD M14.3</p> <p>CRET</p> <p>·</p> <p>·</p> <p>·</p>

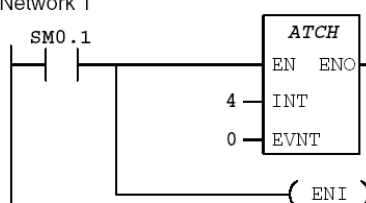

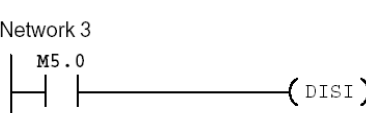
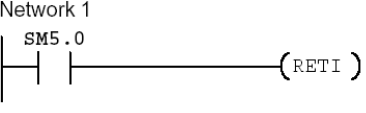
Start of Subroutine 10

A conditional return (RET) from Subroutine 10 may be used.

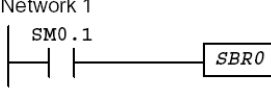
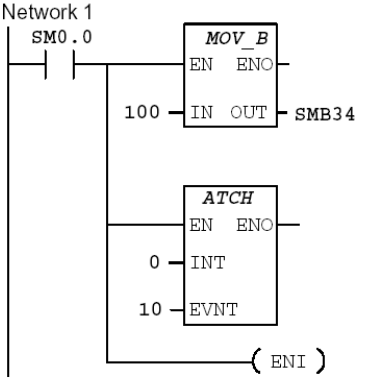
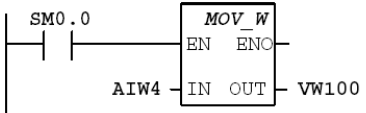
Each subroutine is automatically terminated by STEP 7 Micro/WIN 32 3.0. This terminates Subroutine 10.

Este exemplo demonstra uma subrotina sendo chamada e após sua execução, retorna ao ponto de partida, para a próxima linha do programa.

INTERRUPÇÕES

LAD	STL
MAIN OB1	
<p>Network 1</p>  <p>On the first scan: Define interrupt routine 4 to be a rising edge interrupt routine for I0.0.</p> <p>Globally enable interrupts.</p> <p>Network 2</p>  <p>If an I/O error is detected, disable the rising edge interrupt for I0.0. (This rung is optional.)</p> <p>Network 3</p>  <p>Disable all interrupts when M5.0 is on.</p>	<p>Network 1</p> <pre>LD SM0.1 ATCH 4, 0 ENI</pre> <p>Network 2</p> <pre>LD SM5.0 DTCH 0</pre> <p>Network 3</p> <pre>LD M5.0 DISI</pre>
INTERRUPT 4	
<p>Network 1</p>  <p>I/O rising edge interrupt subroutine. Conditional return based on I/O error End of I0.0 rising edge interrupt routine.</p>	<p>Network 1</p> <pre>LD SM5.0 CRETI</pre>

Configurando uma interrupção

LAD	STL
MAIN PROGRAM	
<p>Network 1</p>  <p>First scan memory bit: Call Subroutine 0.</p>	<p>Network 1</p> <pre>LD SM0.1 CALL 0</pre>
SUBROUTINE 0	
<p>Network 1</p>  <p>Begin Subroutine 0. Always on memory bit: Set timed interrupt 0 interval to 100 ms.</p> <p>Global Interrupt Enable Attach timed interrupt 0 to Interrupt routine 0.</p>	<p>Network 1</p> <pre>LD SM0.0 MOVB 100, SMB34</pre> <p>ATCH 0, 10 ENI</p>
INTERRUPT 0	
<p>Network 1</p>  <p>Begin Interrupt routine 0. Sample AIW4. Terminate Interrupt routine.</p>	<p>Network 1</p> <pre>LD SM0.0 MOVW AIW4, VW100</pre>

Configurando uma Timed Interrupt para ler o valor de uma entrada analógica.

4.Utilizando-se apenas de um elemento temporizador, elabore um programa de PLC capaz de acionar uma lâmpada de sinalização piscante com período de 2 segundos.

5.Elaborar um programa PLC capaz de interromper automaticamente o funcionamento de uma esteira transportadora de peças. A parada se realiza sempre que um sensor ótico não detectar a passagem de uma nova peça num intervalo menor do que 5 segundos. O religamento da esteira se dá pelo comando do operador em uma botoeira. Identifique qual esquema de temporização foi utilizado na solução.

6.Utilizando-se dos recursos de contagem em PLC, elabore um programa capaz de acionar uma lâmpada sinalizadora sempre que o número de pulsos recebidos em sua entrada for múltiplo de 5 (cinco). Assim, no recebimento do quinto pulso a lâmpada acende, desligando-se no sexto pulso; novamente acende no décimo e desliga no décimo primeiro e assim sucessivamente

7.Para os diagramas elétricos indicados abaixo, implemente-os em linguagem ladder e descreva o funcionamento do circuito. Programe-os no STEP-7:

