UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Aluno: Mário de Sousa Araújo Neto **Matrícula:** 20221104

Professor/Orientador: Péricles Rezende Barros

Data: 11/11/2005

Estágio Supervisionado

RELATÓRIO

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

Índice

Introdução

Este trabalho propõe a introdução ao estudo da automação industrial, através de um material didático sobre o controlador lógico programável (CLP) S7-300 da SIEMENS. Foram descritos o CLP, uma das linguagens de programação utilizadas neste dispositivo e mostrados alguns exemplos de implementação.

No primeiro capítulo é mostrado o hardware do CLP, mostrando sua estrutura, os módulos de entrada e saída, os meios disponíveis para a comunicação e os módulos de endereçamento.

O segundo capítulo diz respeito ao software de programação do CLP, o STEP 7, e mostra como configurar uma CPU e a rede onde o CLP estará inserido, e ainda, como criar o programa que irá implementar a atividade de automação.

O terceiro capítulo serve para mostrar como o programa é processado pela CPU e nos mostra a programação em blocos individuais na linguagem de programação ladder (LAD).

O quarto e último capítulo mostra a implementação de pequenos programas na linguagem LAD, na forma de exemplos explicados, e que podem ser implementados diretamente no CLP disponível no Laboratório de Instrumentação Eletrônica e Controle, do Departamento de Engenharia Elétrica, da Universidade Federal de Campina Grande.

Capítulo1 – O hardware do CLP

1. Introdução

A automação industrial exige a realização de muitas funções. A figura 1.1 representa a chamada pirâmide de automação, com os diferentes níveis de automação encontrados em uma planta industrial.

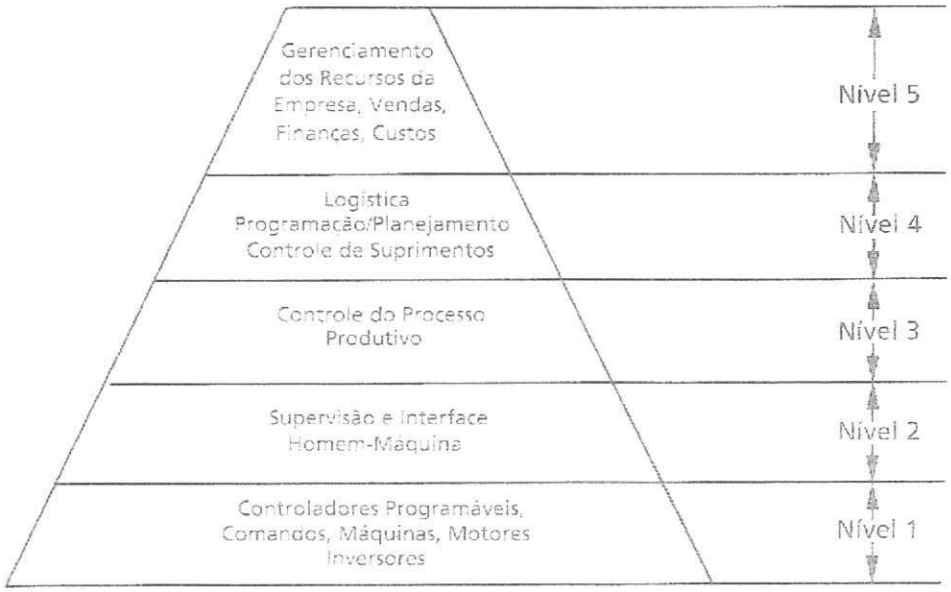


Figura 1.1 Pirâmide de automação mostrando os vários níveis de uma indústria e os componentes da automação.

Na base da pirâmide está frequentemente envolvido o Controlador Lógico Programável (CLP), atuando via inversores, conversores ou sistemas de partida suave, sobre maquinas e motores e outros processos produtivos. No topo da pirâmide, a característica marcante é a informatização ligada ao setor corporativo da empresa.

O nível 1 é o nível das máquinas, dispositivos e componentes (chão de fábrica), onde a automação é realizada pelo CLP. O nível 2 tem como principal característica algum tipo de supervisão associada ao processo. É o nível onde se encontram os concentradores de informações sobre o Nível 1 e as interfaces Homem-Máquina (IHM).

Sob o ponto de vista físico, podemos observar na figura 2 os níveis 1 e 2 da pirâmide de automação.

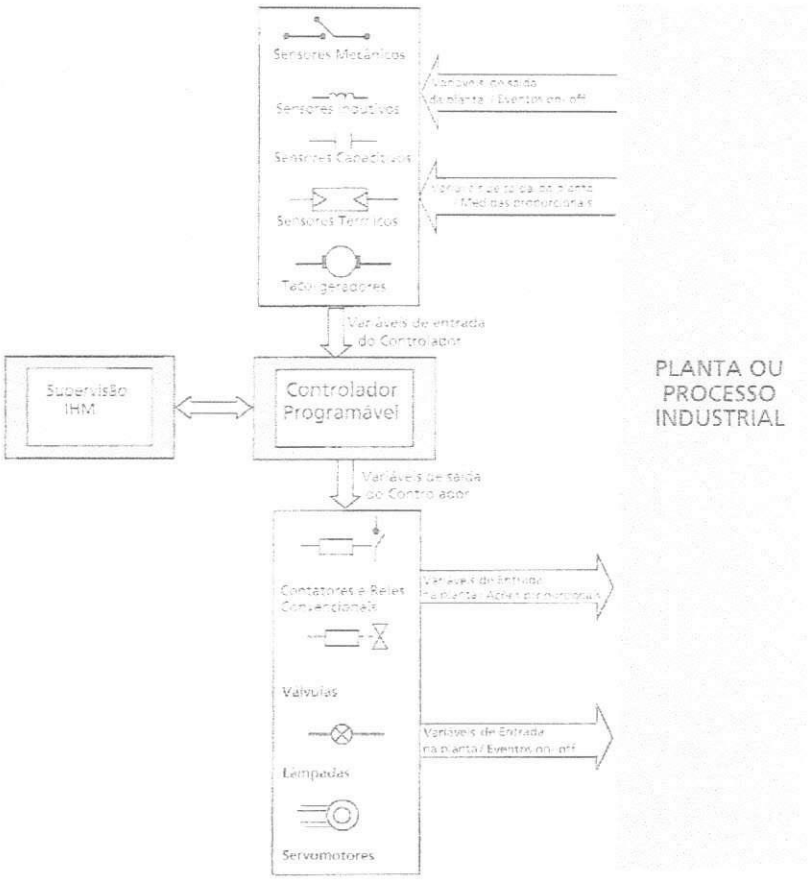


Figura 1.2

Com o Controlador Programável é possível automatizar uma grande quantidade de ações, substituindo o homem, com mais precisão, confiabilidade e rapidez. As informações de entrada são analisadas, as decisões tomadas, os comandos ou acionamentos são enviados às saídas, tudo acompanhando o desenrolar do processo.

1.2. O CLP S7-300

1.2.1 Estrutura do CLP

O S7-300 é um CLP modular composto pelos seguintes componentes:

- **Racks**

  São gabinetes onde os módulos são acomodados e conectados entre si.

- **Fonte de Alimentação (PS)**

  Fornece as tensões necessárias ao funcionamento dos equipamentos.

- **Unidade Central de Processamento (CPU)**

  Armazena e executa o programa de usuário.

- **Módulos de Interface (IMs)**

  Conecta os racks entre si.

- **Sub-redes**

  Conecta o controlador programável a cada um ou a outro dispositivo.

Módulos de I/O

- **Módulos de Sinal (SMs)**

  Adapta os sinais vindos um sistema para os níveis internos de sinal ou atuadores controlados por sinais digitais e analógicos.

- **Módulos de Função (FMs)**

  Executa processos complexos ou de tempo crítico independentemente da CPU.

- **Processadores de Comunicação (CPs)**

  Estabelece a conexão com as redes auxiliares.

Um controlador programável (ou Estação) pode ser composto por mais de um rack conectados entre si por meio de um barramento. A fonte de alimentação, CPU e módulos de I/O são colocados no rack central. Se não houver espaço suficiente ou se desejar adicionar novos módulos, outros racks podem ser conectados ao rack central através dos módulos de interface (IMs).

Em um controlador S7-300, podem ser conectados 8 módulos de I/O no rack central. Caso esta configuração de apenas uma camada seja insuficiente pode-se conectar rack entre si através dos módulos de interface, como nos mostra a figura 1.3.

O número de módulos em um rack pode ser limitado ainda pela corrente máxima através deste, que é de 1.2A.

Os racks são conectados aos módulos através de dois barramentos: o barramento de I/O (ou barramento P) e o barramento de comunicação (ou barramento K).

- **Barramento de I/O** → é destinado à troca rápida de sinais de entrada e saída, sendo útil em trocas que envolvem grande quantidade de dados. Este barramento é também chamado de Barramento P.
- **Barramento de comunicação** → conecta a CPU e o dispositivo de interface de programação (MPI) com módulos de função e processadores de comunicação. Este barramento é também chamado de Barramento K

Figura 1.3 mostra três tipos de configuração: com uma, duas e quatro camadas. Note que na configuração com mais de uma camada é necessário o uso de módulos de interface em cada rack.

1.3 Áreas de memória da CPU

O programa do usuário em si é dividido em duas áreas denominadas *memória de armazenamento* e *memória de trabalho.*

A **memória de armazenamento** pode ser integrada à CPU ou pode ser um cartão de memória. Todo o programa do usuário, incluindo os dados de configuração, fica neste local.

A **memória de trabalho** funciona como uma memória RAM de alta velocidade totalmente integrada à CPU. Nesta área da memória são carregados trechos importantes do programa do usuário.

O dispositivo de programação transfere todo o programa do usuário incluindo os dados de configuração para a memória de armazenamento. O sistema operacional do CPU copia o código relevante do programa e os dados do usuário para a memória de trabalho. Quando o programa for lido de volta para o dispositivo de programação, os blocos são trazidos da memória de armazenamento, complementados pelos valores atuais dos dados dos endereços da memória de trabalho.

A **memória do sistema** contém os endereços que o usuário irá acessar no programa. Os endereços são combinados em áreas contendo um número específico de endereços por CPU. A memória do sistema em uma CPU contém as seguintes áreas de endereçamento:

- Entradas (I)

  As entradas são uma imagem dos módulos de entrada digitais.

- Saídas (Q)

  As saídas são uma imagem dos módulos digitais de saída.

- Bit de memória (M)

  Armazenam informação que é acessível em todo o programa

- Temporizadores (T)

  São áreas usadas para implementar estados de espera e tempo de monitoramento.

- Contadores (C)

  São locais no software usados para contagens crescentes e decrescentes.

- Dados Locais Temporários (L)

São locais usados como "*buffers*" dinâmicos intermediários durante o processamento do bloco.

As letras entre parênteses representam as abreviações que serão utilizadas nos endereços quando se escreve o programa. Também é possível atribuir um símbolo para cada variável e então usá-lo no lugar do identificador de endereço.

1.4 Dispositivos de Entrada e Saída (E/S)

Os dispositivos de entrada e saída são dispostos de forma distribuída e conectados por uma rede chamada PROFIBUS-DP. A PROFIBUS-DP fornece uma interface padrão para a transferência predominantemente de dados binários entre um módulo de interface no controlador programável (central) e os dispositivos instalados em campo.

O módulo de interface é chamado de DP mestre e os dispositivos instalados de DP escravos. E/S distribuídos refere-se aos módulos conectados através da PROFIBUS-DP a um módulo mestre PROFIBUS. O DP mestre e todos os escravos controlados por ele formam o sistema DP mestre.

O **DP mestre** é o nó ativo na rede PROFIBUS. Este troca dados ciclicamente com seus DP escravos. Um DP mestre pode ser:

- Uma CPU com uma interface DP mestre integrada ou um sub-módulo conectado (exemplo CPU 315-2DP);
- Um módulo de interface em conjunto com uma CPU;

Os **DP escravos** são os nós passivos na rede PROFIBUS. No SIMATIC S7, uma distinção é feita entre:

- DP escravos compactos

  Comportam-se como um simples módulo para o DP mestre.
- DP escravos modulares

  Englobam vários módulos (sub-módulos)
- DP escravos inteligentes

  Contêm um programa de controle, o qual controla seus próprios módulos.

Sistema DP mestre

- Sistema mestre simples

A PROFIBUS-DP é geralmente colocada em funcionamento como um sistema mestre simples, ou seja, um DP mestre comandando todos os DP escravos. O DP mestre é o único mestre no barramento, com exceção de um dispositivo de programação temporário. O DP mestre e os DP escravos ligados a ele formam um sistema DP mestre.

- Sistema mestre composto

Pode-se também instalar vários sistemas DP mestres na mesma rede PROFIBUS. De qualquer forma, isto reduz o tempo de resposta individual, pois quando um DP mestre inicializa seus escravos, a permissão de acesso é passada para o próximo DP mestre então este inicializa seus escravos e assim consecutivamente.

- Vários Sistemas DP mestres por Estação

Pode-se reduzir o tempo de resposta se um sistema DP mestre contiver apenas alguns escravos. Desde que seja possível operar vários DP mestres em uma estação S7, consegue-se distribuir DP escravos de uma estação sobre vários sistemas DP mestres. Em modo multiprocessado, toda CPU tem seu próprio sistema DP mestre.

1.5 Comunicações

A Comunicação - troca de dados entre controladores programáveis - é realizada integralmente pelo SIMATIC S7. Quase todas as funções para comunicação são gerenciadas através do sistema operacional. Você pode trocar dados sem nenhum hardware adicional e com apenas um cabo de conexão entre dois CLPs.

O SIMATIC NET é o responsável pela comunicação no SIMATIC. Este componente realiza a troca de informações entre controladores programáveis e entre o controlador programável e o dispositivo de interface homem-máquina.

Uma rede é uma conexão entre dispositivos com o propósito de prover a comunicação. Ela engloba uma ou mais, idênticas ou diferentes, sub-redes conectadas entre si.

Numa sub-rede, todos os pontos de comunicação são interligados via hardware com características físicas uniformes e parâmetros de transmissão uniformes (como por exemplo taxa de transferência de dados) e a troca de dados é feita através de um processo de transmissão comum. O SIMATIC reconhece MPI, PROFIBUS, Ethernet Industrial e conexão ponto a ponto (PTP) como sub-redes.

As Sub-redes são os objetos centrais para a comunicação no SIMATIC Manager, elas diferem entre si em performance

- **MPI**

  Solução de baixo custo para interligar poucos CLPs com baixo fluxo de dados;

- **PROFIBUS**

  Troca rápida de pequeno a médio fluxo de dados. Usado preferencialmente com E/S distribuídos;

- **Ethernet Industrial**

  Estabelece comunicação entre computadores e CLPs para troca rápida de grande fluxo de dados;

- **Ponto a Ponto (PTP)**

  Ligação serial entre dois dispositivos através de protocolos especiais.

1.6 Módulos de endereçamento

O Caminho do sinal

Quando você conecta sua máquina ou planta ao CLP, você determina quais sinais e onde serão conectados ao controlador programável.

Um sinal de entrada, por exemplo, o sinal de uma chave que ativa um motor (Ligar motor), é enviado para o módulo de entrada através do terminal ao qual ele foi conectado. Esse terminal possui um "endereço", chamado endereço de E/S (no caso da figura, byte 5, bit 2).

Antes de qualquer programa iniciar sua execução, a CPU copia automaticamente o sinal para a imagem das entradas do processo, onde então,

esta é acessada como um endereço de entrada (I 5.2, por exemplo). A expressão I 5.2 é o **endereço absoluto**.

Você pode então nomear esta entrada com símbolos alfanuméricos que correspondem, na tabela de símbolos, ao endereço absoluto (como "Motor ligado"), ajudando na identificação. A expressão "Ligar Motor" é um endereço simbólico.

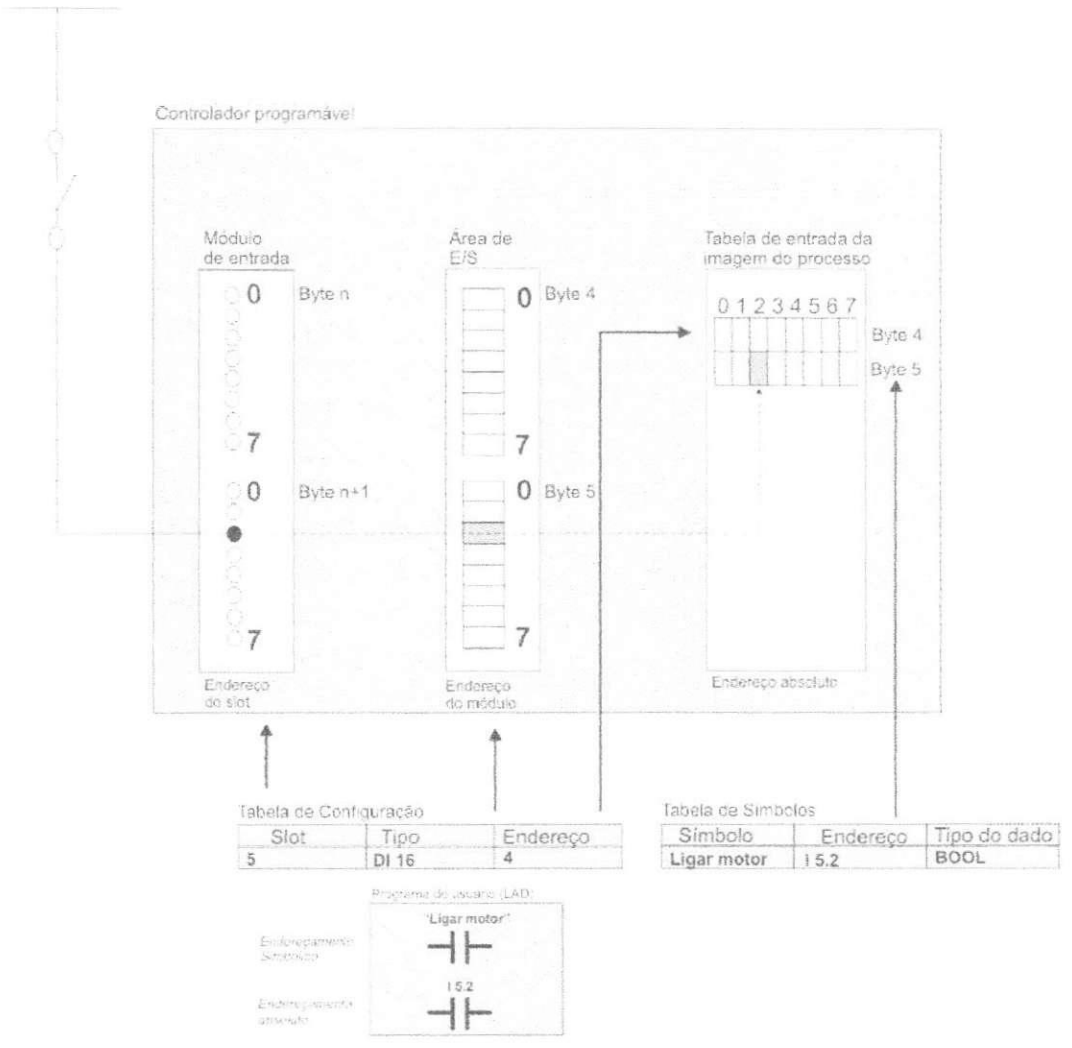


Figura 1.4 Correlação entre Endereço do Módulo, Endereço Absoluto e Endereço Simbólico.

Endereço Físico

Todo slot (abertura no módulo) tem um endereço fixo no controlador programável (uma estação S7). Este endereço consiste no número do rack onde a estação está montada e o número do slot. Um módulo é unicamente descrito usando seu endereço físico

Se o módulo contém cartões de interface, cada um desses cartões também é associado a um endereço de sub-módulo. Dessa forma, cada sinal analógico e digital e cada conexão serial no sistema tem seu endereço único.

Correspondentemente, módulos de E/S distribuídos também possuem endereço físico. Neste caso, o número do DP mestre e o número da estação sobrepõem o número do rack.

A ferramenta usada para configuração de hardware acompanha o programa STEP 7 e é usada para distribuir, de forma física, os módulos através da estação.

Endereço inicial do módulo

Além do endereço físico, que define o slot, cada módulo tem um endereço inicial, que define a localização no espaço de endereçamento lógico (espaço de endereço de E/S). O espaço de endereço de E/S começa no endereço 0 e termina no limite superior específico de cada CPU.

O endereço inicial do módulo determina como os sinais de entrada e saída são endereçados (acessados) pelo programa. No caso de um módulo digital, os sinais individuais (bits) são agrupados em grupos de oito chamados bytes. Estes bytes tem endereços relativos 0, 1, 2 e 3; o endereço dos bytes começam no endereço inicial do módulo. Exemplo: No caso de um módulo digital com quatro bytes e endereço inicial 8, os bytes individuais são acessados pelo endereço 8, 9, 10 e 11. No caso de um módulo analógico, os sinais individuais (tensões e correntes) são chamados de canais, cada um ocupando dois bytes. Módulos analógicos estão disponíveis com 2, 4, 8, e 16 canais, correspondendo a 4, 8, 16 ou 32 bytes de área de endereçamento.

Módulos de E/S distribuídos (estações) reservam um número específico de bytes no espaço de endereçamento de E/S. Os endereços dos módulos centralizados e aqueles dos módulos distribuídos não podem se sobrepor.

O CLP S7-300 é um dos mais utilizados nas plantas industriais. Destacamos sua estrutura modular que é de grande vantagem além de poder ser utilizado em configurações de várias camadas em aplicações de grande porte.

Capítulo 2 - O Software de Programação STEP 7

Este capítulo descreve o pacote básico do software de programação STEP 7 que é utilizado tanto para configurar o hardware e a rede como para escrever o programa que irá executar a tarefa de automação.

1 SIMATIC Manager

O *Simatic Manager* é a ferramenta principal do Step 7, nele você trabalha com objetos "lógicos" que correspondem a objetos reais na sua planta. A forma mais simples de criar um projeto é através do *project* wizzard. Ao abrir o programa, o project wizzard é inicializado.

A primeira tela do wizzard é apenas de apresentação. Clique em next para começar a criação do projeto.

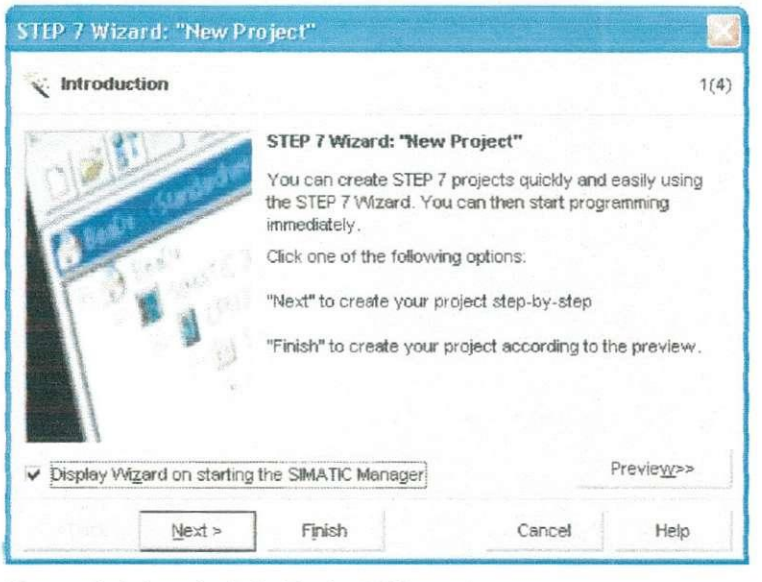


Figura 2.1 Janela 1 do *Project Wizzard*

Na segunda tela, você será perguntado sobre qual é a CPU que está utilizando no seu projeto. O CLP do LIEC é equipado com a CPU315-2DP.

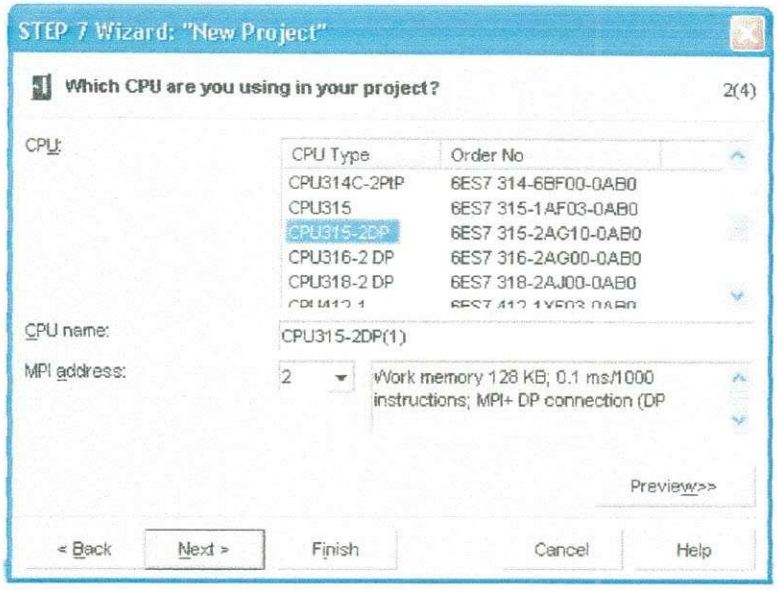


Figura 2.2 Janela 2 do *Project Wizzard*

Na terceira janela será perguntado quais objetos você deseja adicionar ao seu projeto. Basicamente é necessário adicionar o objeto de execução cíclica (OB1) onde o programa principal será escrito, além disso, deve-se informar a linguagem a ser utilizada nesses blocos.

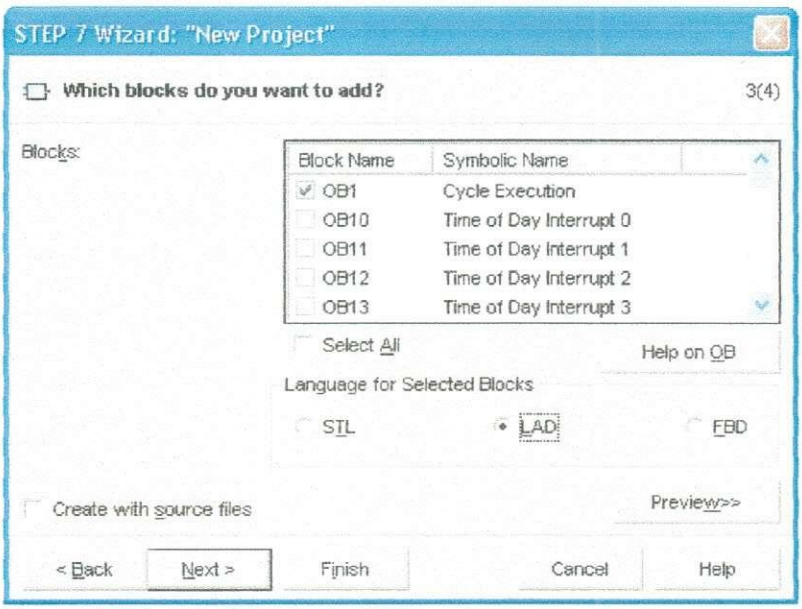


Figura 2.3 Janela 3 do *Project Wizzard*

Na quarta e última tela você deve dar um nome ao projeto e pode ver também os projetos existentes.

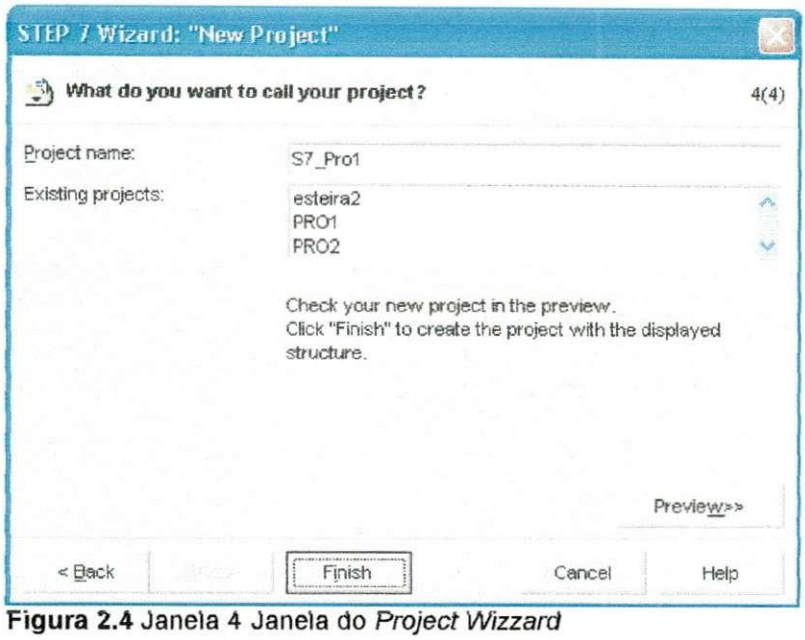


Figura 2.4 Janela 4 Janela do *Project Wizzard*

No projeto, está contida toda a planta industrial, além das estações (CLPs) e os programas necessários para completar uma tarefa de automação. As estações são compostas por uma CPU onde o programa será armazenado e o programa é composto por objetos como o OB1 citado anteriormente.

Também pode-se criar um projeto sem ter que especificar uma CPU. Para isto, deve-se cancelar o project wizzard e criar o projeto manualmente clicando em FILE → NEW... Dê um nome ao projeto e ele será criado. Ao projeto, deve-se então adicionar um "container" do programa S7. Poderá ser programado normalmente desta forma e depois adicionar os outros componentes, como CPU e sub-redes, ao seu projeto.

Aqui, vale associar os "containers" às "pastas" e os "objetos" aos "arquivos" no Windows. Dê um duplo-clique em um container e ele mostrará os objetos que ele contém. Dê um duplo-clique em um objeto e o Simatic Manager inicializará uma ferramenta apropriada para editá-lo.

Os principais componentes de um projeto são:

- Dados de configuração do hardware (CPUs);
- Dados de parametrização dos módulos;
- Dados de configuração da rede de comunicação;
- Programas (códigos, símbolos, fontes)

Com o STEP7, são oferecidas também algumas bibliotecas, utilizadas para armazenar componentes reutilizáveis do programa:

- **Blocos de função do sistema** – Contém as interfaces dos blocos do sistema para programação offline;
- **Blocos de conversão S5-S7 e T1-S7**– Contém as funções para os conversores S5-S7 e T1-S7;
- **Blocos de função IEC** – Contem as funções chamadas para editar as variáveis do tipo DATA_AND_TIME e STRING;
- **Blocos de comunicação** – Contém as funções necessárias para controlar os módulos dos processadores de comunicação (CPs);
- **Blocos de controle PID** – Contém as funções necessárias para o controle em malha fechada;
- **Blocos de organização** – Contém as tabelas para a organização dos dados.

2. Configurando as Estações

A ferramenta de configuração do hardware é utilizada para configurar o CLP, além de endereçar e parametrizar os módulos. Para inicializar a ferramenta de configuração, clique no container SIMATIC 300 e depois duplo-clique no objeto HARDWARE.

Se o projeto foi criado com a ajuda do project wizzard, o rack e a CPU já estarão configurados.

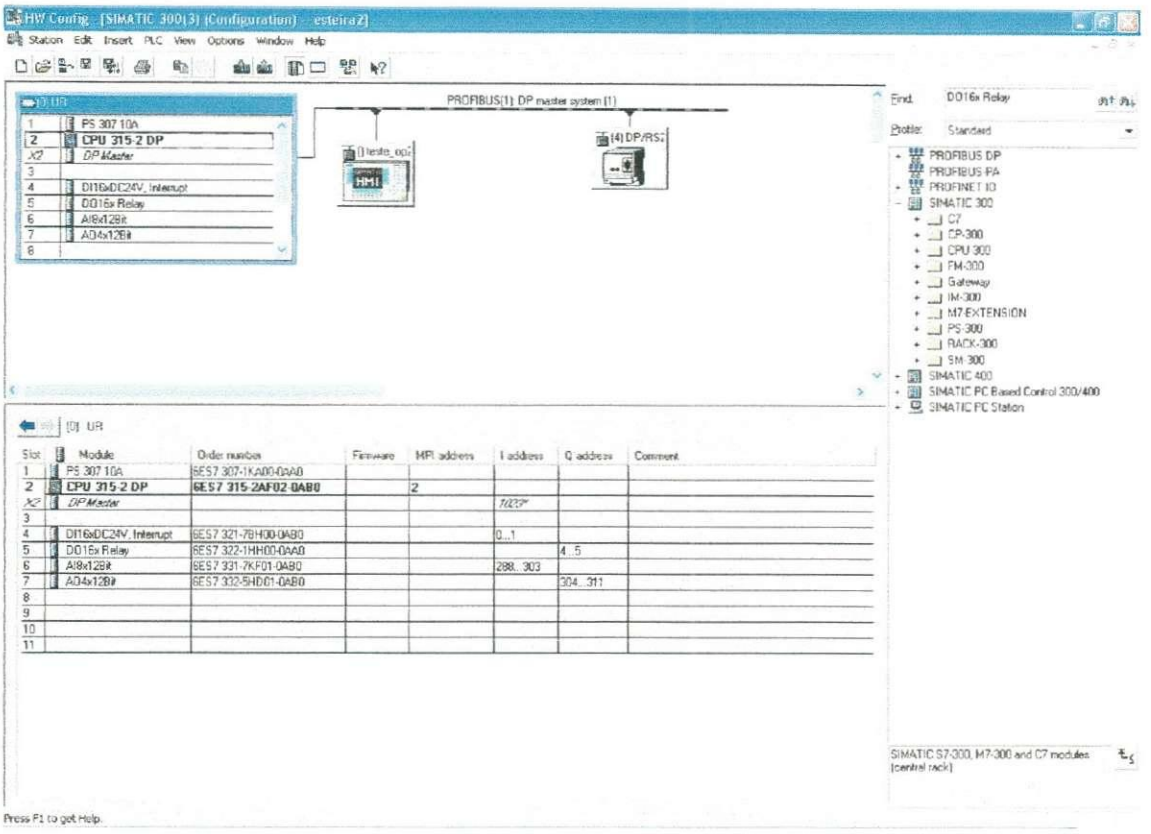


Figura 2.5 Ferramenta para configuração do hardware

A janela da ferramenta de configuração nos mostra no seu lado direito o catálogo do hardware, que contém todos os racks de montagem, módulos e sub-módulos de interface reconhecidos pelo Step7. No canto superior é mostrado, em forma de tabela, o rack de montagem e dentro dele os módulos e sub-módulos alocados no rack. Na parte inferior, é mostrada a tabela de configuração que nos dá detalhes do dispositivo selecionado na parte superior da janela.

Caso não tenha sido adicionada nenhuma ainda, mas deseja adicionar uma nova CPU ao seu projeto, deve fazê-lo clicando no rack desejado, arrastando e soltando em qualquer parte vazia da janela de configuração. Ele será mostrado como uma tabela. Para dentro da tabela, deve-se arrastar a CPU e os outros módulos que deseja adicionar ao rack. A posição 3 da tabela é reservada para o módulo de interface. Poderá ser notado, o endereço inicial é atribuído automaticamente e mostrado na tabela da parte inferior da janela.

As configurações básicas podem ser feitas selecionando o objeto HARDWARE do container SIMATIC300 e clicando em OPTIONS → CUSTOMIZE.

Quando a configuração estiver completa, será necessário checar a consistência dos dados para garantir que não há erros de configuração. Para tanto, vá em STATION → CONSISTENCY CHECK. Garantido que o projeto está livre de erros, salve-o em STATION → SAVE. As configurações serão armazenadas nas tabelas de configuração.

Clicando em STATION → SAVE AND COMPILE, você salva o projeto e compila as tabelas de configuração. Os dados serão salvos no objeto System Data. Uma vez compilado, o projeto pode ser transferido para o CLP clicando em CLP → DOWNLOAD. Para transferir os dados da CPU do CLP para o HD do computador, usa-se CLP → UPLOAD.

3. Configurando a Rede

Quando um projeto é criado, o Simatic Manager cria automaticamente uma Sub-rede MPI. Clicando duas vezes no objeto CONNECTIONS no container CPU, a ferramenta de configuração de rede é iniciada.

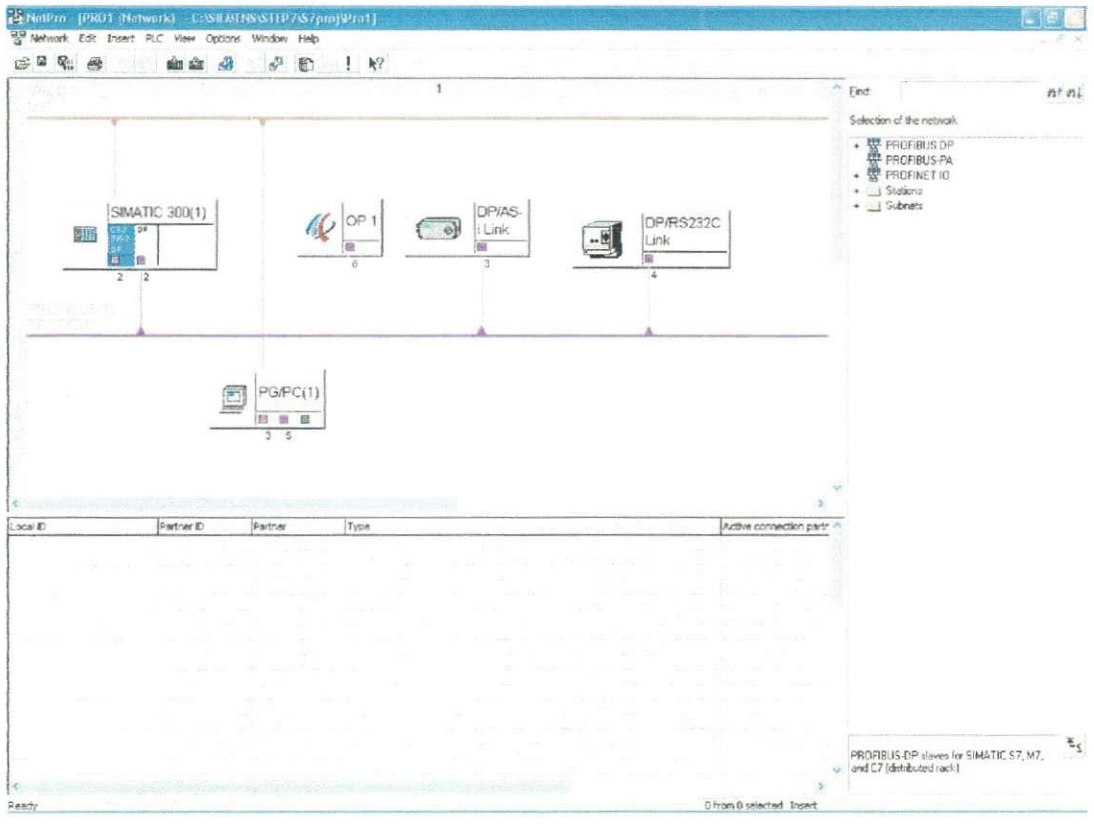


Figura 2.6 Ferramenta para a configuração da rede

Na parte superior da janela, é mostrada a CPU e seus módulos e a rede MPI (barramento vermelho). Se já tiver sido configurada anteriormente, serão mostradas as sub-redes criadas e as estações (nós) com as conexões configuradas.

A tabela de conexões é mostrada na parte inferior da janela se um objeto, capaz de se comunicar, for selecionado na parte superior. No canto direito da janela é mostrado o catalogo de objetos de comunicação.

Da mesma forma como foi feito no utilitário de configuração de hardware, deve ser adicionado sub-redes ou estações clicando e arrastando para a janela. Se um módulo tem capacidade de comunicação, ele fica marcado com um quadrado na cor padrão da rede. Para conectá-lo à sub-rede, basta clicar no quadrado e arrastá-lo até o barramento correspondente. Os parâmetros de configuração podem ser modificados clicando em EDIT → OBJECT PROPERTIES.

4. Criando um Programa S7

O programa do usuário é criado sob o objeto S7_PROGRAM. Este objeto é criado automaticamente dentro do container CPU quando você cria o projeto com o wizzard. Um programa pode ser criado independentemente de se ter adicionado uma CPU, clicando com o botão direito no projeto e dentro de INSERT → NEW OBJECT → S7 PROGRAM. O container S7_Program é criado com os objetos SOURCE FILES, SYMBOLS E BLOCKS. Dentro de BLOCKS você encontrará o objeto OB1, onde ficará o programa principal.

4.1 Tabela de Símbolos

Em um programa S7, o programador irá trabalhar basicamente com endereços para as entradas, saídas, temporizadores, contadores, blocos, etc. Como exemplo, os endereços das entradas serão da forma I0.1 e das saídas Q2.1. Este tipo de endereçamento é conhecido como *endereço absoluto*. Como forma de tornar o programa mais amigável e de facilitar a identificação das variáveis, podem ser atribuídos "nomes" (símbolos) a esses endereços. Neste caso, uma variável pode ser chamada de "Partida do Motor 1". Este tipo de endereçamento é conhecido como *endereço simbólico*.

No modo de endereçamento simbólico, são diferenciados os símbolos *locais* e os símbolos *globais*. Um símbolo local é conhecido somente no bloco onde ele foi definido. Você poderá usar o mesmo símbolo local em diferentes blocos para fins distintos. Um símbolo global é conhecido por todo o programa e tem o mesmo significado em todos os blocos. Estes começam com uma letra e podem ter no máximo 24 caracteres, incluindo caracteres especiais.

O endereço simbólico global pode ser definido na *tabela de símbolos* e pode ser atribuído aos seguintes objetos:

- Entradas (I), Saídas (Q), Entradas Periféricas (PI) e Saídas Periféricas (PQ);
- Bits de memória (M), Temporizadores (T) e Contadores (C);
- Blocos de código OBs e FBs;
- Tipos de dados definidos pelo usuário;
- Tabela de variáveis.

Quando um programa S7 é criado, uma tabela de símbolos vazia é criada. Ao abri-la (EDIT → SYBOLS), você poderá atribuir símbolos globais aos endereços absolutos. Você também deverá especificar o tipo de dado na coluna "Data Type". O tipo de dado define propriedades especificas do dado que o símbolo representa, por exemplo, o tipo BOOL representa uma variável binária, já o tipo INT, representa uma variável digital que contém um inteiro de 16 bits. Adiante serão mostrados outros tipos de variáveis.

Para iniciar o editor do programa, clique duas vezes no objeto OB1 (gerado automaticamente). As opções do editor do programa podem ser modificadas em OPTIONS → CUSTOMIZE .

O programa pode ser escrito tanto on-line como off-line. No modo off-line você escreve o programa e salva o bloco no disco rígido (FILE → SAVE), depois transfere para a CPU clicando em CLP → DOWNLOAD. Para editar um bloco que está na CPU (on-line), primeiro devemos transferi-lo da CPU para o dispositivo de programação onde será editado. Depois de salvo, o objeto pode ser transferido novamente para a CPU e será executado no próximo ciclo.

Um suplemento ao programa oferecido pelo Simatic Manager é a possibilidade de ver a referencia dos dados. Estas referencias podem ser acessadas selecionando o objeto BLOCKS e depois o menu OPTIONS →

REFERENCE DATA → DISPLAY e podem ser utilizadas como base para correção de erros e principalmente como um mapa das variáveis. Esta ferramenta inclui:

- Cruzamento de Referencias: Exibe uma lista mostrando os endereços e blocos contidos no programa, incluindo endereço absoluto, símbolo (se houver), bloco onde o endereço é utilizado, como é utilizado (leitura ou escrita) e a linguagem;
- Atribuições das Entradas/Saidas e bits de memória: exibe uma lista mostrando que bits nas áreas de entrada (I), saída(Q) e memória(M) estão atribuídos no programa;
- Estrutura do programa: mostra todos os níveis das camadas dos blocos do programa;
- Símbolos não utilizados: esta lista mostra os endereços que estão na tabela de símbolos mas não estão sendo utilizados no programa;
- Endereços sem símbolo: mostra os endereços que não tem símbolos associados.

5. Modo On-Line

Criada a configuração do hardware, o programa e estando este armazenado no HD na forma compilada é hora de transferi-lo para a CPU. Para isso, o dispositivo de programação deve ser conectado ao CLP.

Esta conexão é estabelecida entre a interface MPI do dispositivo de programação e a interface MPI do CLP. Quando existe apenas um controlador, a conexão é feita diretamente, mas, se na sub-rede MPI houver várias CPUs, deve ser atribuído um endereço diferente para cada controlador. Para isso, deve-se conectar o dispositivo de programação diretamente a cada CPU e transferir o programa individualmente (PLC → DOWNLOAD). Assim, cada CPU terá seu próprio endereço na sub-rede.

5.1 Testando o Programa

Depois de estabelecer a conexão com a CPU e ter transferido o programa para a sua memória, é hora de testá-lo. Você pode testar o programa por completo ou em partes (cada bloco individualmente). Se a CPU parar de funcionar por conta de um erro, é possível encontrar a causa num processo de depuração.

Para isso, seria necessária a utilização de módulos de simulação para inicializar as variáveis ou executar o programa em um software de simulação como o PLCSIM. É altamente indicado que o programa seja testado antes de ligar a estação na planta, que na maioria das vezes trabalhará com equipamentos de grande porte.

O software de programação STEP 7 é extremamente robusto e oferece muitas funcionalidades ao programador. A Ferramenta "Project wizzard" facilita muito a configuração do projeto. É preciso notar que a rede deve ser configurada após a criação do projeto para que a comunicação do CLP com o terminal de programação seja ativada.

Capítulo 3 – O programa no SIMATIC S7

1. Introdução

Neste capítulo é mostrada a estrutura do programa do usuário para o SIMATIC S7-300/400 desde as diferentes classes de prioridade (tipos de execução de programa) passando pelas partes do programa do usuário (blocos) até as variáveis e tipos de dados. O foco deste capítulo é a descrição da programação em bloco utilizando a linguagem Ladder (LAD).

Define-se a estrutura do programa do usuário antes, na fase de desenvolvimento. Para uma programação eficaz, é necessário dedicar atenção especial à estrutura do programa.

2. Processamento do programa

O programa global de uma CPU consiste no sistema operacional e o programa do usuário.

O *sistema operacional* é o conjunto de todas as instruções e declarações que controlam os recursos do sistema e os processos que usam estes recursos, e inclui o backup dos dados em caso de uma falta de energia, a ativação das classes de prioridade, e assim por diante. O sistema operacional é um componente da CPU que você, como usuário, não tem acesso para fazer nenhuma modificação. No entanto, você pode recarregar o sistema operacional a partir de um cartão de memória, por exemplo, em caso de uma atualização ser necessária.

O programa do usuário é o conjunto de todas as instruções e declarações necessárias ao processamento de sinais pelos quais uma planta (processo) é supervisionado conforme a tarefa de controle definida.

2.1 Métodos de Processamento do programa

O programa do usuário é composto de seções de programa que a CPU processa dependendo de certos eventos. Tal evento poderia ser o inicio do sistema de automação, uma interrupção, ou a detecção de um erro no programa. Os programas correspondentes aos eventos são divididos em classes de prioridade que determinam a ordem de processo do programa (interrupção mútua) quando vários eventos ocorrem.

O programa de menor prioridade é o programa principal que é processado ciclicamente pela CPU. Qualquer outro evento pode interromper o programa principal em qualquer momento, a CPU executa então o serviço da rotina de interrupção ou a rotina de tratamento de erro e retorna ao programa principal.

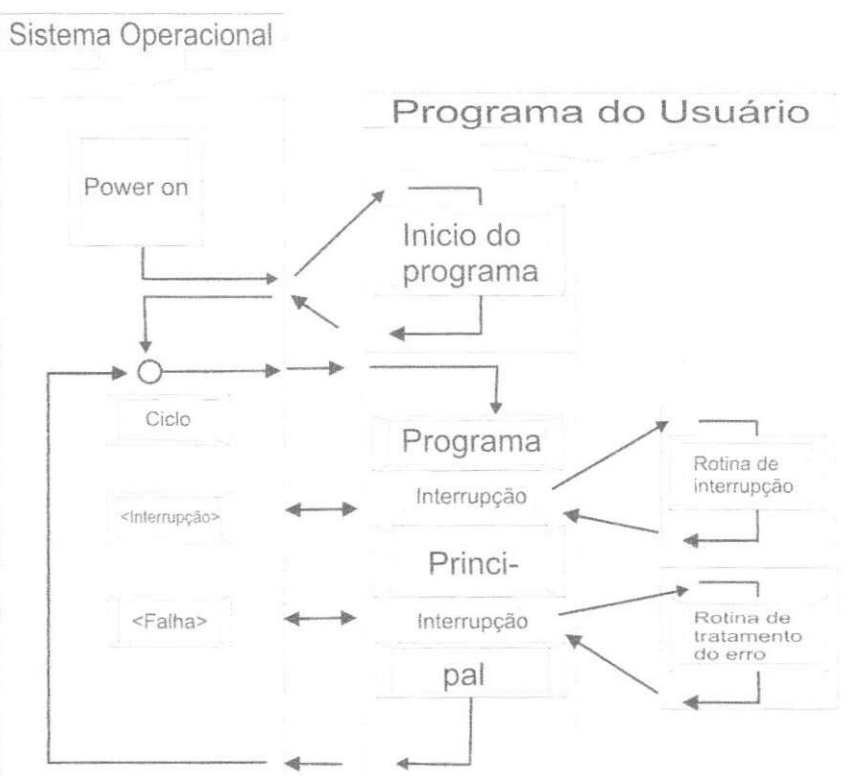


Figura 3.1 Ciclo de execução do programa no controlador programável

Um *bloco de organização* específico (OB) é alocado para cada evento. Os blocos de organização representam as classes de prioridade no programa do usuário. Quando um evento acontecer, a CPU chama o bloco de organização associado. Um bloco de organização é uma parte do programa do usuário que você pode escrever.

Antes de executar o programa principal, a CPU executa uma rotina inicial. O programa principal está no bloco de organização OB 1, e está sempre sendo processado. Depois que OB 1 é processado (fim do programa), a CPU volta ao sistema operacional e, depois de executar várias funções do sistema operacional, como a atualização das imagens do processo, chama OB 1 uma vez mais.

Os eventos que podem interromper o programa são interrupções e erros. Interrupções podem vir do processo (interrupções de hardware) ou da CPU (interrupção cão de guarda, interrupção hora-do-dia, etc.). Quanto aos erros, uma distinção é feita entre erros síncronos e assíncronos. Um erro assíncrono é um erro que é independente da execução do programa, por exemplo, falta de energia para uma unidade de expansão ou uma interrupção gerada porque um módulo estava sendo substituído. Um erro síncrono é um erro causado pelo processamento do programa, como acesso a um endereço não existente ou erro de conversão de dados.

2.2 Classes de prioridade

A tabela 1 mostra os blocos de organização disponíveis no SIMATIC S7, cada com sua prioridade. Em algumas classes de prioridade, você pode mudar a prioridade atribuída quando você configura a CPU. A Tabela mostra as mais baixas e mais altas classes de prioridade possíveis.

Tabela 1 Blocos e suas prioridades

Bloco de organização	Quando é chamado	Prioridade	
		Default	Modificação
Execução livre OB 1	Ciclicamente pelo sistema operacional	1	não
Interrupções TOD OB 1 a OB 17	Em hora específica ou em intervalos regulares	2	2 a 24
Interrupções de atraso OB 20 a OB 23	Depois de um tempo programável, controlado pelo programa do usuário	3 a 6	2 a 24
Interrupções "cão de guarda" OB 30 a OB 38	Regularmente em intervalos programáveis	7 a 15	2 a 24
Interrupções de processo OB 40 a OB 47	Em sinais de interrupção de módulos de E/S	16 a 23	2 a 24
Interrupção de multiprocessador OB 60	Guiado por evento via programa do usuário quando em modo de multiprocessador	25	não
Erros de redundância OB 70, OB 72, OB 73	 Em caso de perda de redundância por erros de E/S Em caso de erro na redundância da CPU Em caso de erro na redundância na comunicação	 25 28 25	 2 a 26 2 a 26 2 a 26
Erros assíncronos OB 80, OB81 a OB 84, 86 e 87, OB 85	Em caso de erros não relacionados com o programa (Ex. erro no relógio, interrupção de diagnóstico, inserir/remover módulo)	26 26 26	26 2 a 26 24 a 26
Processamento em segundo plano OB 90	Duração mínima do ciclo ainda não alcançada	29	não
Rotina de inicialização OB 100, 101 e 102	Ao inicializar o CLP	27	não
Erros síncronos OB 121 e 122	Em caso de erros relacionados à execução do programa (Ex. erro de acesso à E/S)	Prioridade do OBs que causaram os erros	

O bloco de organização OB 90 (processamento em segundo plano) é executado alternadamente com bloco de organização OB 1 , e pode, como OB 1, ser interrompido por todas as outras interrupções e erros.

A rotina inicial pode estar no bloco de organização OB 100 (reinicialização completa) ou OB 101 (reinicialização morna), e tem prioridade 27. Erros assíncronos que acontecem na rotina de inicialização têm prioridade classe 28. Interrupções de diagnóstico são tratadas como erros assíncronos.

Você determina qual das classes de prioridade disponíveis que vai utilizar quando configura a CPU. Blocos de organização não utilizados, devem ter prioridade 0. Os blocos de organização importantes devem ser programados para todas as classes de prioridade usadas, caso contrário a

CPU poderá chamar o OB 85 ("Erro de Processamento do Programa") ou parar.

3. Blocos

O programa pode ser subdividido em tantas seções se desejar, para torná-lo mais fácil de ler e entender. Estas seções do programa são chamadas "Blocos". Um bloco é uma seção do programa do usuário que é definida pela sua função, estrutura ou propósito planejado. Estes blocos são divididos em blocos do usuário e blocos do sistema.

3.1 Blocos do usuário

Em programas extensos e complexos, a subdivisão do programa em blocos é recomendada, e em parte, necessária. Pode ser escolhido entre tipos diferentes de blocos, dependendo de sua aplicação:

Blocos de Organização

Estes blocos servem de interface entre o sistema operacional e o programa do usuário. O sistema operacional da CPU chama os blocos de organização quando eventos específicos acontecem, por exemplo, no caso de uma interrupção de hardware. O programa principal está em um bloco de organização chamado OB 1. Os outros blocos de organização tem números atribuídos de acordo com os eventos que eles são chamados para controlar.

Blocos de função (FBs)

Estes blocos são partes do programa cuja chamada pode ser programada pelos parâmetros do bloco. Eles têm uma memória variável que fica alocada no bloco de dados.

Funções (FCs)

As funções são utilizadas para programar atividades que ocorrem freqüentemente. Elas podem ser configuradas, e devolvem um valor (chamado valor da função) para o bloco de onde foi chamada. O valor da função é opcional, além do valor da função, funções também podem ter outros parâmetros de saída. As funções não armazenam informação.

Blocos de dados (DBs)

Estes blocos contêm os dados do seu programa. Programando os blocos de dados, você determina de que forma os dados serão armazenados (em qual bloco, em que ordem, e sob que forma de dados).

3.2 Blocos do sistema

Blocos do sistema são componentes do sistema operacional. Eles podem conter programas (funções do sistema (SFCs) ou blocos de função do sistemas (SFBs)) ou dados (bloco de dados do sistema (SDBs)). Blocos do sistema tornam várias funções acessíveis ao usuário, como manipular o relógio interno da CPU, ou várias funções de comunicações. Você pode chamar SFCs e SFBs, mas você não pode modificá-las, nem programá-las.

3.3 Estrutura dos blocos

Essencialmente, os blocos são divididos em três partes:

- **Cabeçalho** que contém as propriedades do bloco, como nome do bloco;
- **Seção de declaração** que contém a declaração das variáveis locais do bloco;
- **Seção do programa** que contém o programa e os comentários do programa.

3.4 propriedades dos blocos

As propriedades dos blocos estão no cabeçalho do bloco. Você pode modificá-las clicando em FILE → PROPERTIES. A aba "General – Part 2" mostra a alocação de memória para o bloco em bytes.

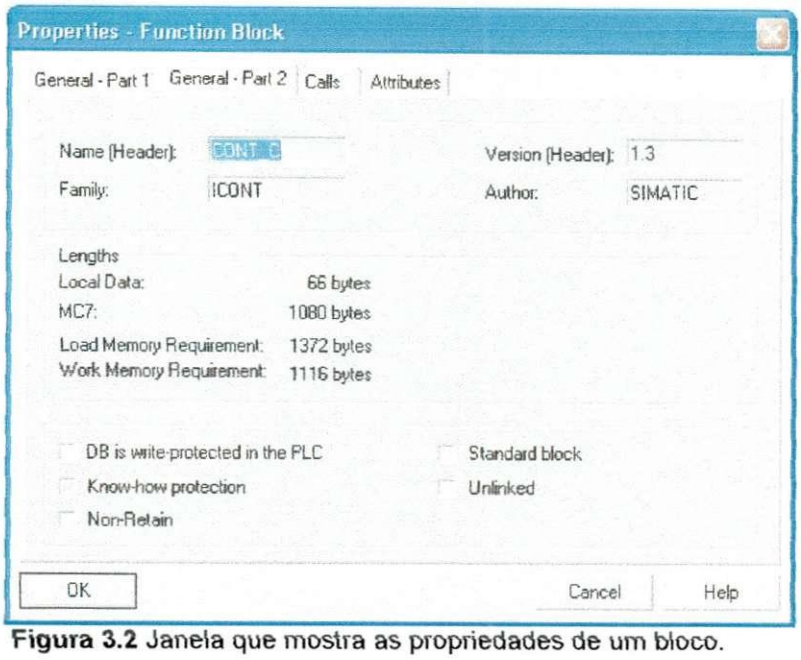


Figura 3.2 Janela que mostra as propriedades de um bloco.

O atributo *KNOW HOW PROTECTION* é utilizado para proteger o bloco. Se um bloco é protegido por este atributo, o programa contido nele não pode ser visualizado, impresso ou modificado. É mostrado apenas o cabeçalho e a tabela de alocação com os parâmetros do bloco. Depois de compilado, ninguém (nem o programador) poderá ter acesso ao conteúdo do bloco

Os blocos marcados com *Standard blocks* são aqueles blocos que vem da Siemens.

O atributo "*DB is write-protected in the PLC*" permite que você leia o bloco de dados. Uma mensagem de erro previne que dados sejam escritos neste bloco.

Um bloco que tenha o atributo *Unlinked* está alocado na memória de carga e sua execução não é relevante. Blocos de dados que estão na memória de carga não podem ser escritos e só podem ser lidos com o auxilio da função SFC 20 BLKMOV.

4. Programando Blocos

Para programar um bloco, você clica nele duas vezes na janela do projeto no SIMATIC Manager. Se o bloco ainda não existe, você deve criá-lo de uma das duas maneiras seguintes:

- Na janela do SIMATIC Manager selecione o objeto BLOCKS e crie um novo bloco clicando em INSERT → S7 BLOCKS → ... . Você verá uma

janela mostrando as propriedades do bloco. Na aba "General – Part 1", selecione o número do bloco e a linguagem "LAD" ou "FBD". Os outros atributos podem ser configurados depois.

- No Editor, clique em FILE → NEW, uma janela irá aparecer, nela você deve escolher o tipo do objeto e o nome. Depois de fechar esta janela, você pode começar a programar.

Você pode dar as informações do cabeçalho do bloco no momento em que o está criando, ou depois no Editor, abrindo o bloco e clicando no menu FILE → PROPERTIES. Quando um bloco é aberto, três janelas são mostradas:

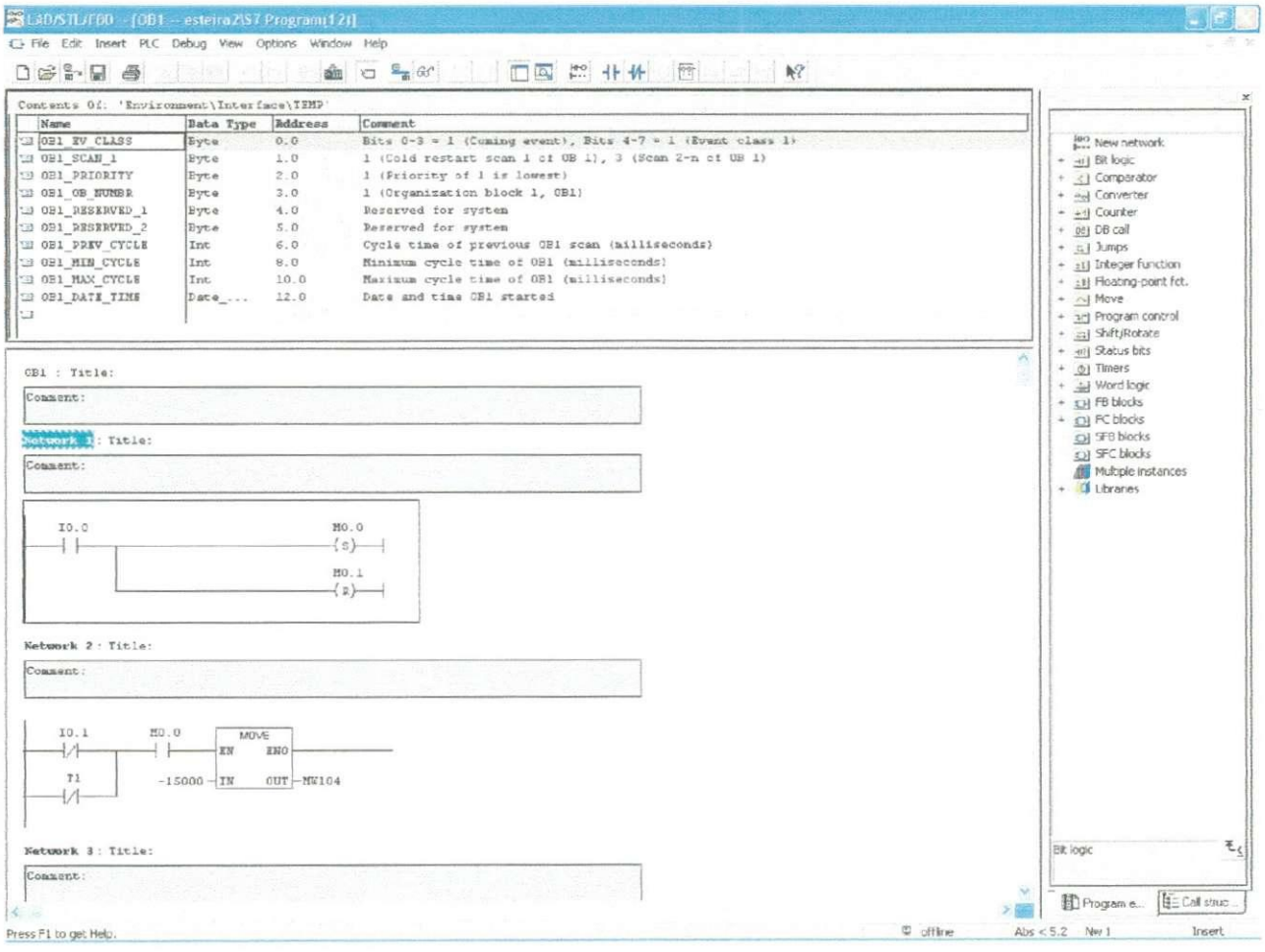


Figura 3.3 Janela do Editor de programas

- **Tabela de declaração de variáveis**, no topo, é onde você define as variáveis locais do bloco;
- A **janela do programa** é onde você vai escrever o programa;
- O **catálogo de elementos** que contém os blocos e funções disponíveis.

Um programa escrito em linguagem de relé (ladder), pode ser subdividido em networks, onde cada um representa um caminho de corrente ou uma operação lógica. O Editor enumera os networks automaticamente, começando do 1 até o 999. Você pode dar a cada network um nome e fazer um comentário.

Para entrar o código do programa, clique uma vez na parte que fica embaixo do network comment. Você verá uma na janela um espaço reservado por uma borda, comece a construir seu programa nesta área. Para inserir um novo network, clique em INSERT → NETWORK.

4.1 Elementos do programa LAD

Os programas em linguagem ladder evoluíram a partir dos diagramas ladder elétricos, que representam a maneira como a corrente elétrica circula pelos dispositivos, de forma a completar um circuito elétrico. Estes diagramas mostram a interconexão entre os dispositivos elétricos em um formato gráfico de fácil leitura, que orienta o técnico na instalação da fiação.

Em um diagrama elétrico, os símbolos representam os dispositivos reais e a maneira com estão conectados. O programa do CLP utiliza símbolos semelhantes só que aqui eles representam instruções lógicas para a aplicação. Um programa em linguagem ladder existe apenas no software do clp: ele não considera a barra de alimentação nem o fluxo de corrente através dos circuitos.

Uma outra diferença é que, em um diagrama elétrico, descreve-se os dispositivos como abertos ou fechados (desenergizados ou energizados). Em um programa ladder, as instruções são Verdadeiras ou Falsas.

Cada linha de um programa ladder deve necessariamente conter, no mínimo, uma instrução de controle (saída), sendo que normalmente contém

uma ou mais instruções de condição (entradas). As condições são programadas à esquerda da instrução de controle.

Entre os exemplos de instruções de condição estão os sinais enviados por dispositivos de entrada que tenham sido conectados, contatos associados a saídas e sinais enviados por temporizadores e contadores.

Programada do lado direito da linha, a instrução de controle é a operação ou função que é ativada/desativada pela lógica da linha. Os exemplos de instruções de controle incluem a energização da saída (de forma a acionar um dispositivo de campo) e as instruções internas ao CLP, tais como comandos de bits, temporizadores, contadores e comandos matemáticos.

As instruções de controle são energizadas ou desenergizadas com base no estado das instruções de condição de linha. Para isto, o CLP examina uma linha quanto à sua continuidade lógica (ou seja, quando todas as instruções de condição são Verdadeiras). Se existir uma continuidade lógica, o CLP energiza a instrução de controle. Se não existir continuidade lógica, o CLP mantém a instrução de controle no estado Desligado ou Desenergizado.

As instruções mais freqüentemente utilizadas em um programa LAD são as instruções "Normalmente Aberto (NA)" e "Normalmente fechado (NF)", que são a representação gráfica de contatos NA e NF, e a instrução "Energizar saída" que é a representação gráfica de uma bobina. As bobinas de Set e Reset funcionam em conjunto e implementam as seguintes funções: Quando a bobina Set é energizada, o endereço ao qual ela está relacionada permanece com valor "1" até que a bobina Reset, associada a este endereço, seja energizada levando seu valor a "0".

Figura 3.4 Representação gráfica dos contatos e bobinas em linguagem Ladder.

No STEP 7 as funções mais complexas são representadas por caixas. Estas funções são os contadores, temporizadores, funções de memória, MOVE, funções aritmética, etc. Ainda é dada a função de uma "caixa vazia" na qual você poderá colocar a função desejada.

Algumas restrições são aplicadas pelo Editor LAD, uma delas é que o ramo mais acima do network, que começa diretamente na barra de força (lado esquerdo), deve terminar com a bobina. Além disso, todos os elementos LAD podem ser colocados neste degrau.

Outras restrições dizem que nenhum elemento LAD deve ser curto-circuitado com um ramo vazio em paralelo e que a "corrente" não pode fluir de um elemento da direita para a esquerda.

5. Variáveis

Uma variável é um valor com um formato específico. Esta consiste de um endereço (como entrada 5.2) e o tipo de dado (como BOOL para um valor binário). O endereço engloba um identificador de endereço (I para entrada) e a localização absoluta de armazenamento da variável (como 5.2 para Byte 5, bit 2). Uma variável também pode ser referida simbolicamente atribuindo um nome (símbolo) na tabela de símbolos.

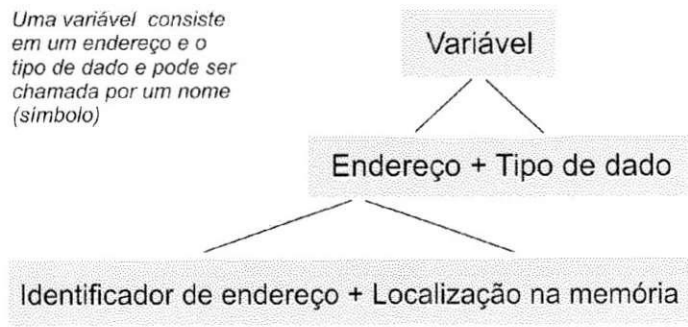


Figura 3.5 Estrutura de uma variável

Um bit de dado do tipo BOOL é chamado de endereço binário (ou operando binário). Endereços que englobem um, dois ou quatro bytes ou variáveis com tipos de dados relevantes são chamados *operandos digitais*.

As variáveis que você declarar dentro de um bloco, são chamadas de variáveis locais. As variáveis locais incluem os parâmetros do bloco, os dados estáticos e temporários. Quando estas variáveis são do tipo elementar, elas

também podem ser acessadas como operandos. As variáveis locais também podem ser do tipo de dado complexo (como estruturas ou arrays)

5.1 Endereçando variáveis

Ao endereçar uma variável, você deve escolher entre endereçamento absoluto e endereçamento simbólico, como foi discutido anteriormente. O endereço absoluto utiliza endereços numéricos começando no zero para cada área de endereço. Endereçamento simbólico utiliza nomes, que você pode definir na tabela de símbolos para o endereçamento global, ou na seção de declaração para os endereços locais. Uma extensão do endereço absoluto é o endereço indireto, onde os endereços dos locais de memória não são computados até que o programa seja executado.

Endereçamento Absoluto

O endereço absoluto de uma entrada ou saída é calculado a partir do endereço inicial do módulo que você estabelece (ou estabeleceu) na tabela de configuração e do tipo de conexão do sinal no módulo. Uma diferença é feita entre sinais analógicos e binários.

Sinais binários: Um sinal binário é aquele que contém apenas um bit de informação. Exemplos de sinais binários são sinais de entrada (chaves) e sinais de saída (contactores).

Sinais analógicos: Um sinal analógico contem 16 bits de informação e corresponde a um "canal" que é mapeado no controlador como uma palavra de 2 bytes. A entrada de sinais analógicos (como voltagem de um termopar) é carregada nos módulos de entrada analógica, digitalizada e então, torna-se disponível ao controlador como 16 bits de informação. Reciprocamente, 16 bits de informação podem controlar um indicador (de pressão, por exemplo) por um módulo de saída analógico, onde a informação é convertida em um valor analógico (como a corrente).

A informação sobre o tamanho de um sinal corresponde ao tamanho da variável onde o sinal é armazenado e processado. A largura de uma informação, associada à interpretação da mesma, produz o *tipo de dado*. Sinais

binários são armazenados nas variáveis do tipo BOOL e sinais analógicos em variáveis do tipo INT.

O único fator determinante para o endereçamento das variáveis é o tamanho da informação. No STEP 7, existem quatro tamanhos que podem ser acessados através do endereçamento absoluto:

- 1 bit Dados do tipo BOOL;
- 8 bits Dados do tipo BYTE ou outro tipo de dado com 8 bits;
- 16 bits Dados do tipo WORD ou outro tipo de dado com 16 bits;
- 32 bits Dados do tipo DWORD ou outro tipo de dado com 32 bits.

Variáveis do tipo BOOL são acessadas por um identificador: o número de um byte e, separado por um ponto, o número de um bit. A numeração dos bytes começa em zero e vai até um limite específico para cada CPU. A numeração dos bits vai de 0 a 7. Exemplos:

I1.0 Bit de entrada nº 0 do byte nº 1

Q16.4 Byte de saída nº 4 do Byte nº 16

Variáveis do tipo BYTE têm como endereço absoluto o identificador de endereço e o número do byte que contém a variável. O identificador é completado por um B.

Exemplos:

IB 2 Byte de entrada nº 2

QB 18 Byte de saída nº 18

Variáveis do tipo WORD são compostas por dois bytes (uma palavra). Elas têm como endereço absoluto o identificador de endereço e o número do bit de menor ordem da palavra que contém a variável. O identificador é completado por um W.

Exemplos:

IW 4 Palavra de entrada nº 4; contém os bytes 4 e 5

QW 20 Palavra de saída nº 20; contém os bytes 20 e 21

Variáveis do tipo DWORD consistem em quatro bytes (uma dupla palavra) que têm como endereço absoluto o identificador de endereço e o

número do byte de menor ordem da palavra que contem a variável. O identificador é seguido por um D.

Exemplos:

ID 8 Dupla palavra de entrada nº 8; contém os bytes nº 8, 9, 10 e 11

QD 24 Dupla palavra de saída nº 24; contém os bytes nº 24, 25, 26 e 27

Endereçamento Simbólico

Utiliza nomes que você mesmo escolhe (chamados de símbolos) em lugar de endereços absolutos. Este nome deve começar com uma letra e ter no máximo 24 caracteres. O STEP 7 diferencia letra maiúscula de minúscula. Uma diferenciação é feita entre símbolos globais e símbolos locais ao bloco.

5.2 Símbolos Globais

Você poderá atribuir nomes na tabela de símbolos aos seguintes objetos:

- Blocos de dados e de códigos
- Entradas, saídas, entradas periféricas e saídas periféricas
- Bits de memória, temporizadores e contadores
- Tipos de dados do usuário
- Tabelas de variáveis

Um símbolo global pode ainda incluir espaços e caracteres especiais, e, neste caso, devem ser escritos entre aspas. Nos blocos já compilados, o editor de programas sempre mostra símbolos globais entre aspas.

5.3 Símbolos locais

Os nomes para os dados locais são especificados na seção de declaração do bloco e podem conter apenas letras, números e o caractere sublinhado (_). Os símbolos locais são válidos somente no bloco onde foram definidos, o que significa que o mesmo símbolo (o mesmo nome da variável) pode ser utilizado num contexto diferente em outro bloco.

O editor de programa mostra símbolos locais começando com "#", quando não, você mesmo deve digitar quando for atribuir o nome.

6. Tipos de dados

O tipo do dado diz respeito ao conteúdo de uma variável e o seu tamanho permitido. O STEP 7 tem alguns tipos de dados pré-definidos que você pode combinar com aqueles que você definir.

6.1 Dados do tipo elementar

As variáveis do tipo elementar podem armazenar um bit, um byte, uma palavra ou uma dupla palavra. As variáveis elementares são descritas abaixo.

BOOL, BYTE, WORD, DWORD e CHAR

Uma variável do tipo BOOL representa o valor de um bit, por exemplo I 1.0. Variáveis do tipo BYTE, WORD E DWORD são seqüências de bits com 8, 16 e 32 bits respectivamente. Uma variável do tipo CHAR representa um caractere em ASCII.

INT

Uma variável do tipo INT armazena uma palavra. Os bits 0 a 14 representam a posição dos dígitos do número e o bit 15 o sinal. Sinal "0" significa que o número é positivo, sinal "1" significa que o número é negativo. Um número negativo é representado como complemento de 2. A faixa de valores que representam um número INT é de 32767 até -32768.

DINT

Uma variável do tipo DINT armazena uma dupla palavra. Um inteiro é armazenado como um DINT quando seu valor excede 32767 ou cai abaixo de -32768 ou ainda quando é precedido do indicador L#. Os bits de 0 a 31 representam a posição dos dígitos do numero e o bit 31 o sinal ("0" positivo e "1" negativo). O número negativo é representado como complemento de 2.

REAL

Uma variável do tipo real representa uma fração e é armazenada como um número de 32-bits em ponto flutuante. Um número inteiro é armazenado

como um real quando tem um ponto decimal e um zero. Para representar um número em potencia de 10, utiliza-se um "e" ou "E" seguido de um número ou fração.

S5TIME

A variável do tipo S5TIME é utilizada para setar os temporizadores nas linguagens LAD, STL e FBD. A hora é especificada em horas, minutos, segundos e milissegundos. Exemplos:

S5TIME#500ms

S5TIME#2h46m30s

DATE

Uma variável do tipo DATE é armazenada em uma palavra e seu conteúdo corresponde à uma data desde 01/01/1990. Sua representação é um ano, mês e dia separados por hífen.

Exemplos:

DATE1990-01-01

D#2168-12-31

TIME

Uma variável do tipo TIME tem como conteúdo uma dupla palavra e contém informações sobre dias (d), horas (h), minutos (m), segundos (s) e mili-segundos (ms).

Exemplos:

TIME#24d20h31m23s647ms

T#21s

T#2.25h

TIME_OF_DAY

Contém uma dupla palavra e representa o tempo decorrido desde que o dia começou (0:00).

Exemplos:

TIME_OF_DAY#00:00:00

TOD#12:00:00

6.2 Dados do tipo complexo

As variáveis do tipo complexo só podem ser declaradas nos blocos de dados como dado local temporário ou como parâmetro do bloco.

DATE_AND_TIME

O tipo de dado DATE_AND_TIME representa a hora e a data do dia. Você pode abreviar o tipo por DT

STRING

O tipo de dado STRING representa uma seqüência de caracteres com até 254 caracteres. Você define o número máximo de caracteres da variável em colchetes que você digita depois da palavra chave STRING. Se não for definido um tamanho, o editor utiliza o valor de 254 bytes.

ARRAY

Um dado do tipo ARRAY representa uma seqüência ou campo com um número fixo de elementos do mesmo tipo. Você especifica o tamanho do campo nos colchetes que seguem o tipo de dado ARRAY. O valor inicial deve ser menor ou igual ao valor final e os dois devem ser inteiros de -32768 a +327 67. Um campo deve ter no máximo 6 dimensões e seus limites separados por uma virgula. Abaixo segue alguns exemplos de declaração de um ARRAY.

Tabela 2 Declaração de variáveis do tipo array

Nome	Tipo	Valor Inicial	Comentários
Valor medido	ARRAY[1..24]	0.4, 1.5, 11, (2.6, 3.0)	Variável array com 24 elemento do tipo REAL
TOD	ARRAY[-10..10]	21 (TOD#08:30:00)	Array de 21 variáveis do tipo TOD
Resultado	ARRAY[1..24, 1..4]	96 (L#0)	Array de duas dimensões com 96 elementos
Char	ARRAY[1..2, 3..4]	2 ('a'), 2 ('b')	Array de duas dimensões com 4 elementos

STRUCT

Os dados do tipo STUCT representam uma estrutura de dados que consiste em um número fixo de componentes que podem ser de tipos de dados diferente.

Você especifica os componentes individuais da estrutura e os tipos dos dados deles na linha abaixo da linha com o nome variável e a palavra chave STRUCT. Podem ser usados todos os tipos de dados incluindo outras estruturas. Um exemplo de declaração de estrutura é mostrado abaixo:

Tabela 3 Declaração de variáveis do tipo *struct*

Nome	Tipo	Valor inicial	Comentário
ContMotor	STRUCT		Estrutura de variáveis simples com 4 componentes
On	BOOL	FALSE	Variável ContMotor.On do tipo BOOL
Off	BOOL	TRUE	Variável ContMotor.Off do tipo BOOL
Atraso	S5TIME	S5TIME#5s	Variável ContMotor.Atraso do tipo S5TIME
VelMax	INT	5000	Variável ContMotor.VelMax do tipo INT
	END_STRUCT		

Entender como o programa é executado pelo CLP é muito importante para uma programação eficaz. A facilidade de se dividir o programa em blocos é uma característica marcante do software SIMATIC S7, pois possibilita maior organização do programa.

Capítulo 4 – Exemplos de aplicações

Introdução

Neste capítulo serão apresentados alguns exemplos de programas em ladder utilizando funções básicas. Para isto será utilizado além de circuitos conhecidos, um sistema de automação instalado no Laboratório de Instrumentação Eletrônica e Controle – LIEC que consistem em uma esteira com uma estufa.

Exemplo 1

Este exemplo mostra a implementação de uma Porta OU utilizando a linguagem ladder. Este circuito é um arranjo em paralelo.

Diagrama Lógico:

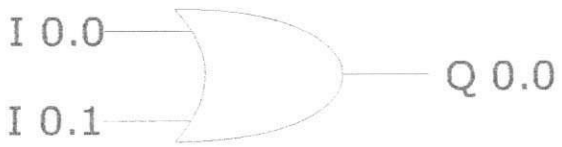
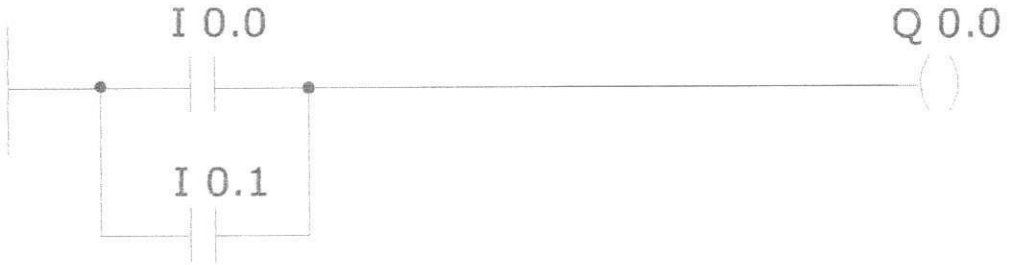


Tabela da Verdade:

I 0.0	I 0.1	Q 0.0
0	0	0
0	1	1
1	0	1
1	1	1

Diagrama Ladder:



Exemplo 2

Este exemplo mostra a implementação de uma Porta AND utilizando a linguagem ladder. Este circuito é um arranjo em série.

Diagrama Lógico:

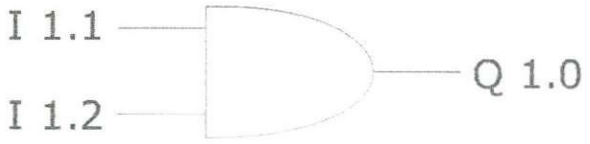


Tabela da Verdade:

I 1.1	*I 1.2*	*Q 1.0*
0	0	0
0	1	0
1	0	0
1	1	1

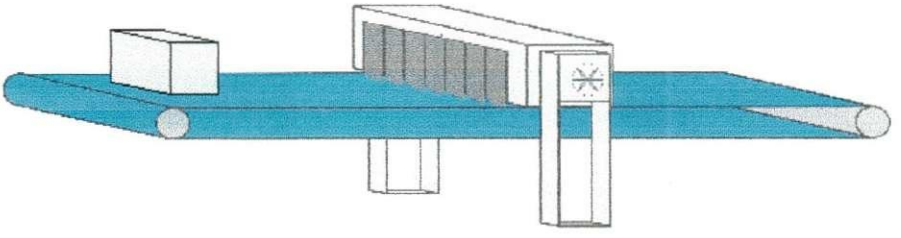
Diagrama Ladder:

Exemplo 3

Este exemplo nos mostra um sistema de controle de uma esteira e nos ilustra o uso da função *MOVE*. A estufa não tem funcionalidade neste exemplo.

Descrição do funcionamento

Quando um produto for colocado no inicio da esteira, deve ser transportado de uma ponta à outra da fábrica.



Sinais e símbolos

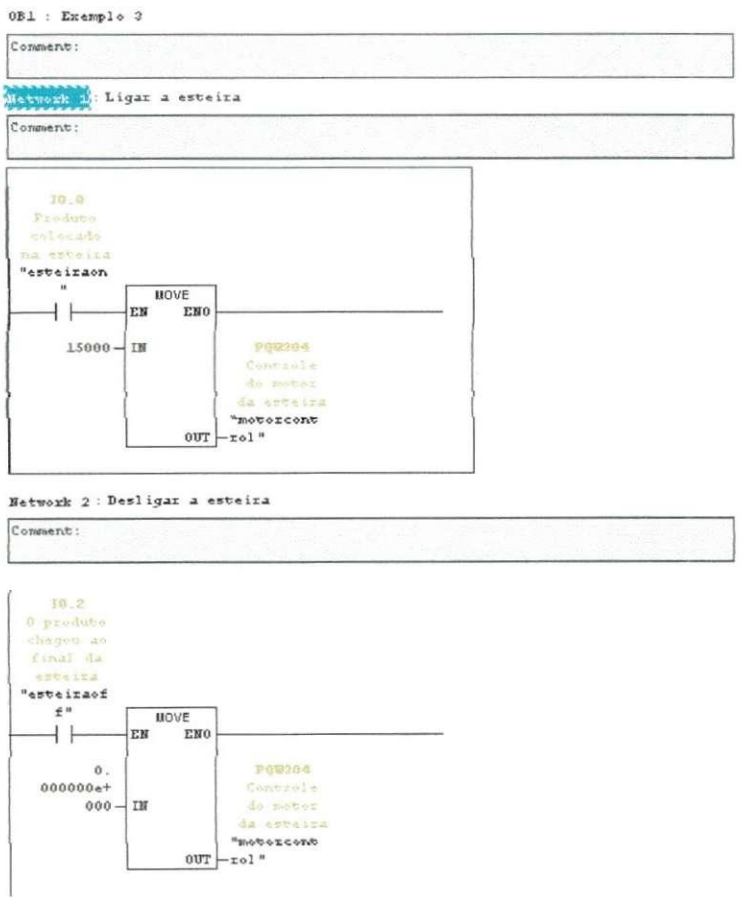
Neste exemplo utilizaremos as entradas I 0.0 e I 0.1. Estas entradas estão ligadas a sensores de presença colocados nas duas extremidades da esteira. Para controlar o motor que movimenta a esteira, utilizaremos a saída PQW 304 que é uma saída do CLP. Note que a variável PQW 304 é uma variável de saída (Q) do tipo palavra (W).

S7 Program(1) (Symbols) -- esteira2\SIMATIC 300(3)\CPU 315-2 DP

	Status	Symbol	Address	Data type	Comment
1		CONT_C	FB 41	FB 41	Continuous Control
2		CONT_S	FB 42	FB 42	Step Control
3		esteiraoff	I 0.2	BOOL	O produto chegou ao final da esteira
4		esteiraon	I 0.0	BOOL	Produto colocado na esteira
5		motorcontrol	PQW 304	WORD	Controle do motor da esteira
6		PULSEGEN	FB 43	FB 43	Pulse Generation
7		SCALE	FC 105	FC 105	Scaling Values
8					

Os símbolos CONT_C, CONT_S, PULSEGEN e SCALE, não são utilizados neste exemplo.

Programa LAD



Exemplo 4

Implementar um programa de controle para a esteira que pegue a peça na divisão de pintura da linha de produção de uma fábrica e entregue à divisão de montagem.

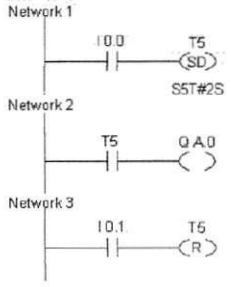
Descrição do funcionamento

A peça chega na esteira pintada com a tinta fresca e deve seguir até a estufa onde passará 10 segundos secando. Após a secagem, a peça volta para a análise de que está ok. Após a análise, a peça segue para a divisão de montagem do outro lado da esteira.

Sinais e símbolos

Entradas / Saídas	Comentários
I 0.0	Sensor de presença do inicio da esteira
I 0.1	Sensor de presença da estufa
I 0.2	Sensor de presença do final da esteira
T1	Temporizador acionado com a chegada do produto na estufa
T2	Temporizador acionado com a chegada do produto novamente ao inicio da esteira
M 0.1, 0.2 e 0.3	Bits de memória

A palavra auxiliar MW104, guarda o valor que será transferido para a saída ligada ao motor (PQW304 – saída periférica tipo palavra) no final do ciclo de execução. Os temporizadores T1 e T2 são do tipo SD e funcionam da seguinte maneira:

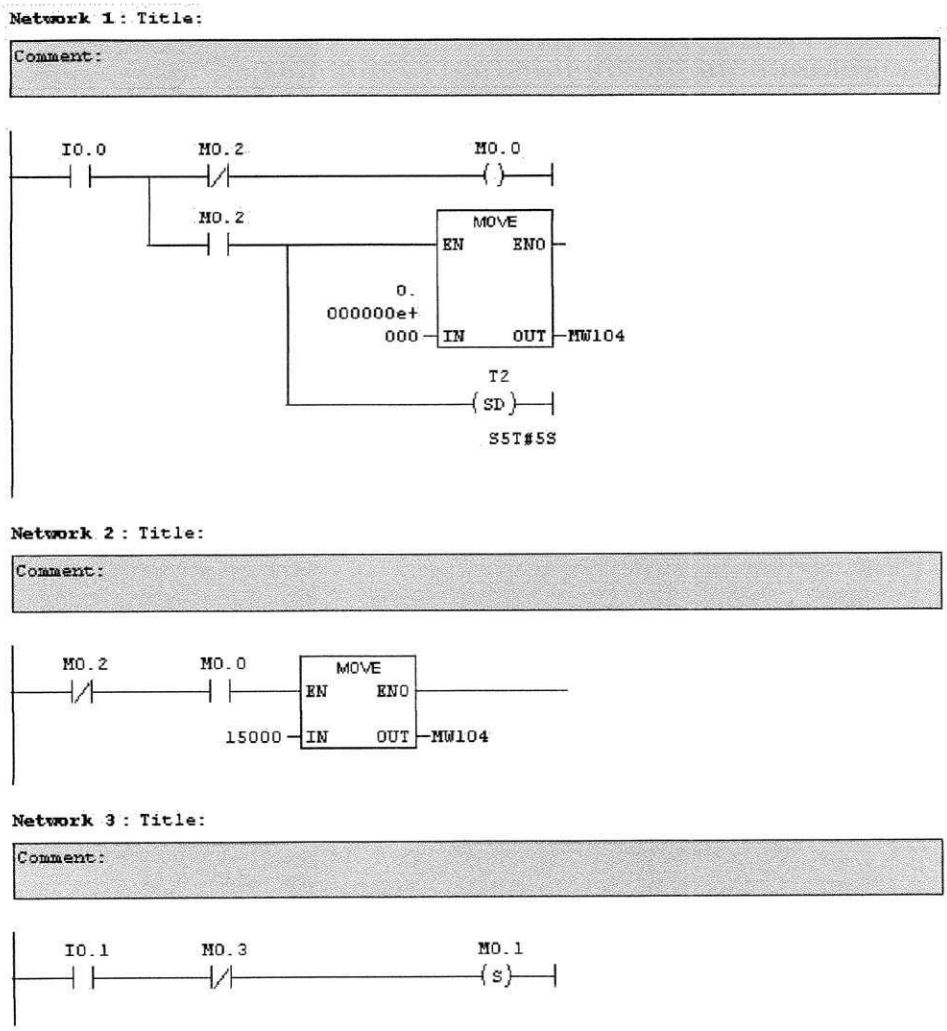


Quando acontece um pulso positivo (passar do estado "0" para o estado "1") na chave I 0.0, o temporizador é iniciado. Se, ao final do tempo especificado (2 segundos) o estado ainda for "1", o estado da saída Q 4.0 será "1".

Quando a chave I 0.1 passar do estado "0" para "1", o temporizador ficará ocioso e o estado da saída Q 4.0 será "0". Se este pulso acontecer enquanto o temporizador está contando o tempo, ele será reiniciado e seu valor vai a zero.

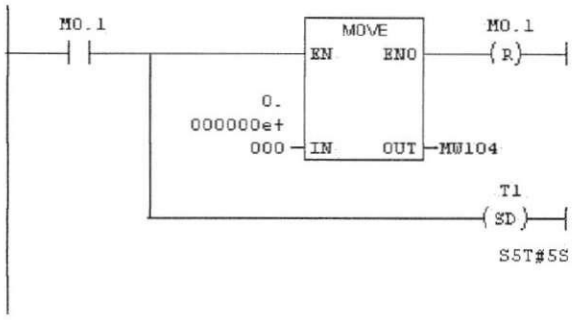
Existem também no programa os bits de memória que são contatos auxiliares os quais não tem nenhuma relação com os objetos da planta ou processo. Os bits auxiliares são muito úteis visto que nem sempre uma saída real está disponível.

Programa em LAD



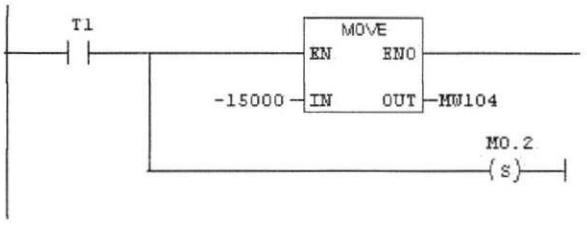
Network 4 : Title:

Comment:



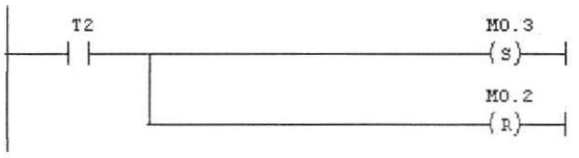
Network 5 : Title:

Comment:



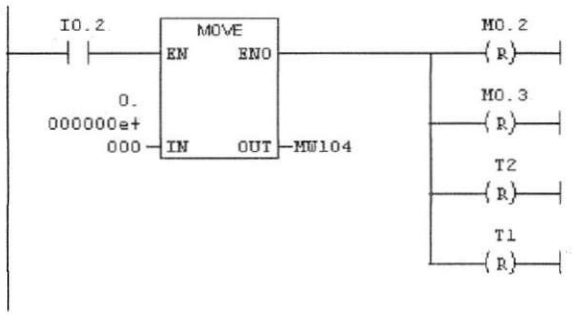
Network 6 : Title:

Comment:



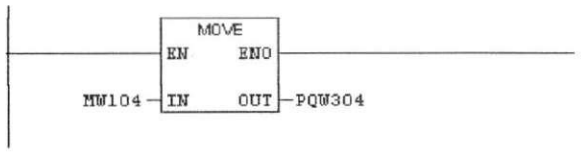
Network 7 : Title:

Comment:



Network 8 : Title:

Comment:



A partir dos exemplos apresentados, pudemos conhecer um pouco mais da estrutura do programa em LAD além do uso de algumas funções descritas neste trabalho. No exemplo 3, é mostrado como um símbolo definido aparece no programa.

Considerações finais

A automação na indústria decorre de necessidades tais como: maiores níveis de qualidade de conformação e de flexibilidade, menores custos de trabalho, menores perdas materiais e menores custos de capital; maior controle das informações relativas ao processo, maior qualidade das informações e melhor planejamento e controle da produção.

Para atender estas necessidades, são de extraordinária importância os CLPs, que tornam a automação industrial propriamente dita uma realidade, pois permitiram reduzir os custos dos materiais, de mão-de-obra, de instalação e de localização de falhas, além de reduzir as necessidades de fiação e os erros associados.

Nesse contexto, o estudo dos CLPs torna-se obrigatório ao futuro profissional da engenharia, visto que toda e qualquer indústria que possua um mínimo de automação em qualquer de seus processos possui hoje sistemas comandados por este dispositivo.

Referências Bibliográficas

- BERGER, Hans – "Automating with STEP 7 in LAD and FBD" – Publicis MCD Corporate Publishing – 2ª Edição – Erlangen e Munich, 2001;

- Rockwell Automation - "Micro Mentor. Entendendo e Utilizando os Microcontroladores Programáveis" - Allen Bradley Company, Inc – 1996;

- MORAES e CASTRUCCI – "Engenharia de Automação Industrial" – LTC – 1ª Edição – Rio de Janeiro, 2001;

- S7 300 Instructions List – Manual em formato eletrônico da Siemens – 3ª Edição;

- S7 300 Programable Controller Instalation and Hardware – Manual em formato eletrônico da Siemens – 2a Edição;

- Home Page da Siemens: www.automation.siemens.com – Consultado em outubro de 2005.