A Norma IEC 61131

A criação de diversos modelos de equipamentos dedicados à automação industrial gerou uma grande variedade de equipamentos e como conseqüência uma incompatibilidade das características com referência a programação dos mesmos.

Para atender às demandas da comunidade industrial internacional, foi formado um grupo de trabalho dentro da International Electrotechnical Commission (IEC) para avaliar o projeto completo de controladores lógicos programáveis, incluindo hardware, instalação, testes, documentação, programação e comunicação.

Algumas forças tarefas de especialistas foram então estabelecidas para desenvolver as diferentes partes do padrão. A força-tarefa 3 recebeu o objetivo primário de desenvolver um novo padrão de linguagens de programação de controladores programáveis, este foi o primeiro esforço internacional efetivo para a padronização das linguagens de programação para a automação industrial. A tabela 1 apresenta o estado atual da norma IEC 61131, que normaliza o projeto de controladores lógicos programáveis:

Parte	Título	Conteúdo	Publicação
Parte 1	General Information	Definição da terminologia e conceitos.	2003 (2ª Ed.)
Parte 2	Equipment requirements and tests	Teste de verificação e fabricação eletrônica e mecânica.	2003 (2ª Ed.)
Parte 3	Programmable Languages	Estrutura do software do CLP, linguagens e execução de programas.	2003 (2ª Ed.)
Parte 4	User guidelines	Orientações para seleção, instalação e manutenção de CLP's.	2004 (2ª Ed.)
Parte 5	Communications	Funcionalidades para comunicação com outros dispositivos.	2000 (1ª Ed.)

Parte 6	Reservada		
Parte 7	Fuzzy Control Programming	Funcionalidades de software, incluindo blocos funcionais padrões para tratamento de lógica nebulosa dentro de CLP's.	2000 (1ª Ed.)
Parte 8	Guidelines for the Application and Implementation of Programming Languages	Orientações para implementação das linguagens IEC 1131-3.	2003 (2ª Ed.)

Tabela 5.1. Partes da norma IEC 61131

Atualmente, apenas algumas especificações e testes de certificação estão disponíveis para esta norma, fazendo com que o número de produtos certificados seja modesto. Mas, já é perceptível a tendência dos fabricantes em atender às características da parte 3 da norma IEC 61131.

IEC 61131 Parte 3: Padronização Internacional de Linguagens, estrutura de software e execução de programas em CLPs

A figura a seguir apresenta a linha do tempo descritiva da evolução dos esforços de normalização e de definição de linguagens de programação de CLPs, a começar pelo ano de 1970, um ano após a instalação do primeiro CLP em uma fábrica nos Estados Unidos.

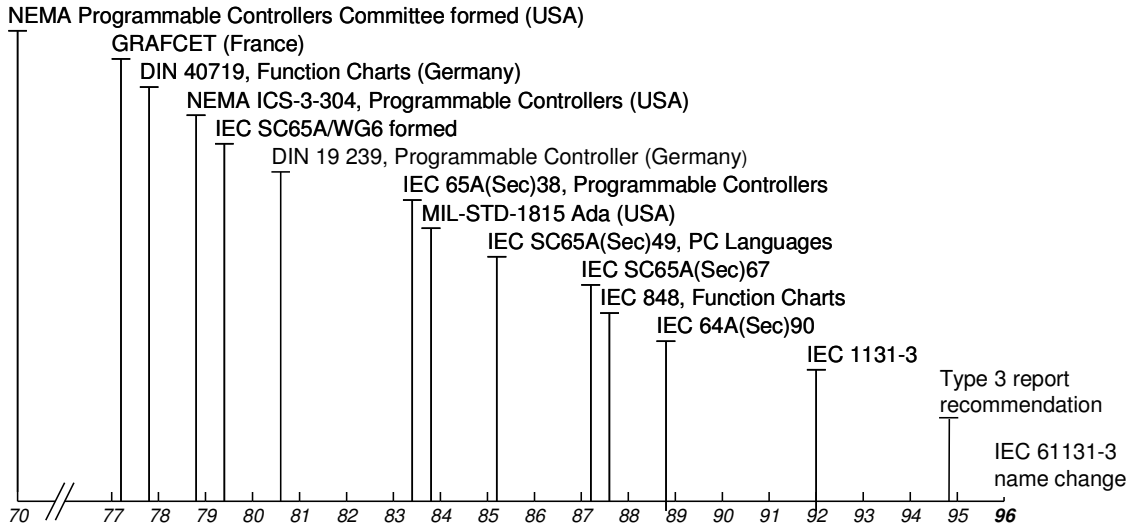


Figura 5.2. Evolução da padronização das linguagens de programação.

Linguagens de Programação na IEC 61131

Existem cinco tipos básicos de linguagem que normalmente são encontradas em controladores programáveis e são padronizadas pela norma IEC 61131-3:

Linguagens Textuais

- Texto Estruturado (Strutured Text – ST)
- Lista de Instruções (Instruction List – IL)

Linguagens Gráficas

- Diagrama Ladder (LD)
- Diagrama Blocos Funcionais (Function Block Diagram – FBD)

Dentro dos elementos comuns definidos pela norma existe o Seqüenciamento Gráfico de Funções ou Sequential Function Chart (SFC).

O SFC descreve graficamente o comportamento seqüencial de um programa de controle e é derivado das técnicas de modelagem por Redes de Petri e da norma IEC 848 que define o padrão Grafcet.

A partir destes dois padrões, o SFC possui alterações necessárias para se viabilizar a conversão de um modelo com representação padrão em um conjunto de elementos de controle de execução adequados a projetos de automação.

O SFC consiste de passos, interligados com blocos de ações e transições. Cada passo representa um estado particular do sistema sendo controlado. Cada elemento ou programa nesta linguagem pode ser programado em qualquer linguagem textual ou gráfica IEC, incluindo-se o próprio SFC quando disponível na ferramenta de programação em questão.

Devido a sua estrutura geral ser mais adequada à representação global de projetos de automação e de programas de grande porte, o SFC funciona também como uma ferramenta de comunicação entre equipes de projetistas, integrando pessoas de diferentes formações, departamentos e países. A figura a seguir exemplifica um trecho de código em SFC:

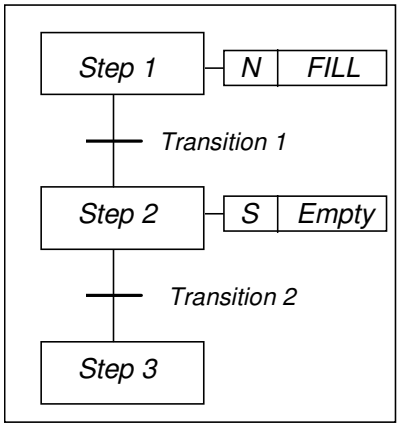


Figura 5.3. Sequenciamento Gráfico de Funções (SFC)

Linguagens Textuais

Texto Estruturado (Strutured Text – ST)

É uma linguagem de alto nível muito poderosa, com raízes em Pascal e “C”. Contém todos os elementos essências de uma linguagem de programação moderna, incluindo condicionais (IF-THEN-ELSE e CASE OF) e iterações (FOR, WHILE e REPEAT).

Exemplo:

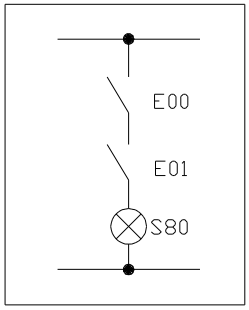
```
I:=25;
WHILE J<5 DO
        Z:= F(I+J);
END_WHILE

IF B_1 THEN
        %QW100:= INT_TO_BCD(Display)
ENDIF

CASE  TW OF
        1,5:    TEMP := TEMP_1;
        2:      TEMP := 40;
        4:      TEMP := FTMP(TEMP_2);
ELSE
        TEMP := 0;
        B_ERROR :=1;
END_CASE
```

Figura 5.4. Texto Estruturado (ST)

Lista de Instruções (Instruction List – IL)

Consiste de uma seqüência de comandos padronizados correspondentes a funções. Assemelha-se a linguagem Assembler.

O programa representado pela linguagem descritiva “Se as entradas E00 e E01 estiverem ligadas, então ligar saída S80” Pode ser representado em lista de instruções por:



	A E00	: Contato E00
AND	A E01	: **EM SÉRIE** Contato E01
	= S80	: = Acionamento de saída S80

Linguagens Gráficas

Diagrama Ladder (LD)

A linguagem Ladder é, conforme mencionado anteriormente, a linguagem de programação de PLCs mais comum e a mais difundida, é também conhecida como lógica de diagrama de contatos, pois se assemelha à tradicional notação de diagramas elétricos e de painéis de controle a relés. O mesmo esquema elétrico apresentado no exemplo anterior pode ser representado em diagrama Ladder por:

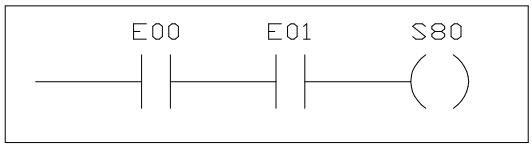


Figura 5.5. Diagrama Ladder (LD)

Diagrama de Blocos Funcionais (Function Block Diagram – FBD)

O diagrama funcional é uma forma gráfica de representação de instruções ou comandos que devem ser executados. É baseado em blocos funcionais, por exemplo, uma porta AND. Estes blocos são em geral utilizados dentro de lógicas ladder. O programa representado pela linguagem descritiva “Se as entradas E00 e E01 estiverem ligadas, então ligar saída S80” pode ser representado em blocos funcionais por:

Figura 5.6. Diagrama de Blocos Funcionais (FBD)

Estrutura de Software e Execução de Programas na IEC 61131

Existem elementos comuns da norma que são usados por todas as linguagens de programação IEC 61131 citadas. Um dos aspectos mais importantes da programação de qualquer sistema é a capacidade de decompor o software em partes menores.

A norma prevê que sejam desenvolvidos ambientes de programação capazes de decompor programas complexos em diferentes elementos de software, os quais possuem uma interface padronizada e bem definida entre os mesmos. O modelo de software (*software model*) consiste em um conjunto de conceitos que definem uma infra-estrutura para decomposição em partes do projeto de automação.

A programação baseada na norma IEC é orientada para o desenvolvimento de programas tanto a partir da abordagem de cima para baixo (top-down) como de baixo para cima (botton-up).

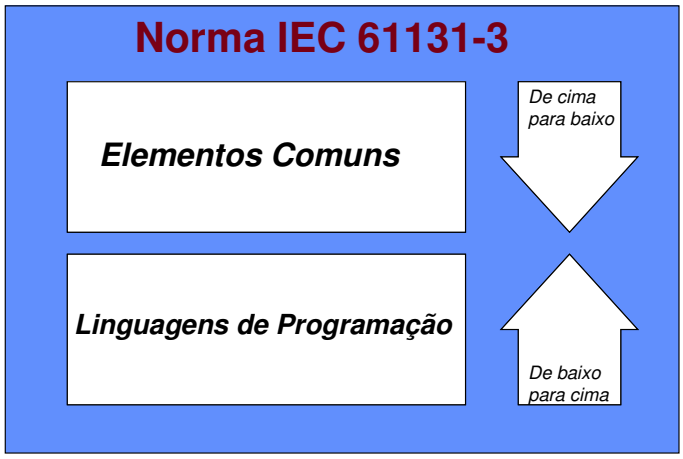


Figura 5.7. Desenvolvimento up-down e top-down.

Utilizando-se o modelo proposto pela norma ISA 88 utilizada em sistemas de controle de batelada, para a modularização de processos industriais em combinação com as estruturas definidas pelo modelo de software da IEC61131, pode-se utilizar a estrutura na figura seguinte para uma melhor compreensão do problema:

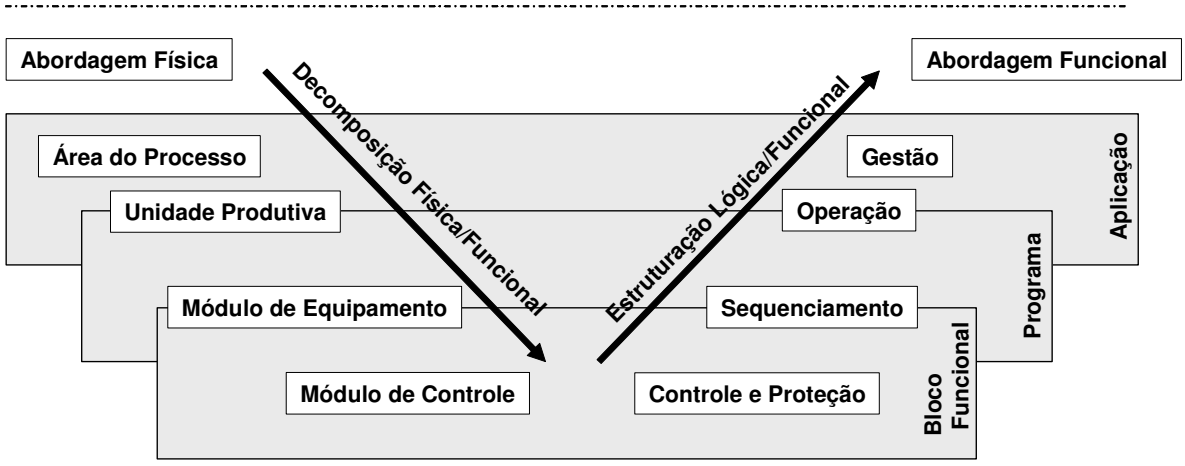


Figura 5.8. Modelo de projeto hierárquico adaptado da norma ANSI/ISA 88

Neste modelo, a linguagem adotada pela ISA 88 para a programação de sistemas de batelada é a SFC.

Modelo de Software

Considerando-se o contexto de um programa de um Controlador Lógico Programável, todo o programa tem que evoluir e interagir com o ambiente onde está

inserido. Não é possível, portanto, se definir a estrutura de um programa de CLP sem um bom entendimento das interfaces e interações com o sistema de controle e com a planta externa.

Conforme mencionado anteriormente, quando um CLP está em modo de execução (modo run), são necessárias as seguintes interfaces:

- **Interfaces de Entrada/Saída:** permite o acesso aos dispositivos ou cartões de E/S para leitura dos sinais do processo como: pressões, níveis, temperaturas, etc..., assim como fazer o comando dos atuadores.
- **Interfaces de comunicação:** muitos sistemas necessitam trocar informações com outros CLP's, IHM's, etc.
- **Interfaces de sistema:** corresponde à interface entre o programa do CLP e o hardware do mesmo, a fim que ocorra um correto funcionamento.

A figura a seguir apresenta o modelo de software definido pela IEC 61131-3:

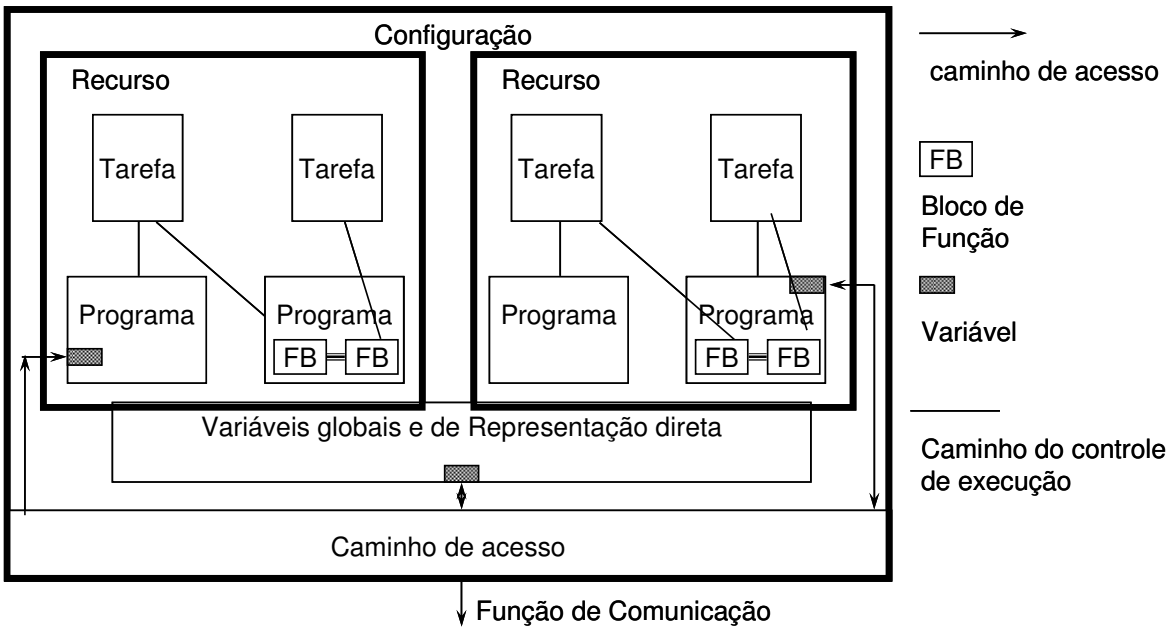


Figura 5.9. Modelo de software IEC 61131-3

Configuração

No nível mais alto, o software para um sistema de controle está contido em uma configuração. Uma configuração corresponde ao software necessário para um único CLP. Porém, em sistemas mais complexos podem existir diversas configurações ou uma configuração com diversos CLP's, as quais interagem entre si através de interfaces de comunicação padronizadas pela norma.

Configuração no Âmbito da norma IEC não deve ser confundida com o termo normalmente usado para expressar os passos para definição de parâmetros e setups para um sistema, ou seja, "configurando um sistema".

Recurso

Dentro de cada configuração podem existir um ou mais recursos. Um recurso é basicamente qualquer elemento com capacidade de processamento, responsável pela execução dos programas. Uma característica dos recursos é que eles são uma divisão de software, entretanto também podem refletir uma divisão no hardware. Cada recurso deve ser independente, não podendo ser influenciado por outros recursos.

Programas

Um programa IEC pode ser construído a partir de diferentes elementos de software escritos em qualquer uma das diferentes linguagens da norma.

Tipicamente, um programa consiste de um código de execução, capaz de trocar dados através das conexões de software. Um programa pode acessar as variáveis do CLP e se comunicar com outros programas. A execução de diferentes partes de um programa pode ser controlada usando *Tasks*.

Segundo a norma, "*programa é uma montagem lógica de todos os elementos de linguagens e construções necessárias para o processamento de sinal requerido para o controle de uma máquina ou processo por um sistema controlador programável".*

Blocos Funcionais

O conceito de blocos funcionais é um dos mais importantes da norma IEC61131-3, para permitir o projeto de software de forma hierárquica e estruturada. blocos funcionais podem ser utilizados para a criação de elementos de software totalmente reutilizáveis, desde a criação de outros blocos funcionais mais simples, até programas mais complexos.

As principais características dos blocos funcionais são que estes possuem um conjunto de dados, os quais podem ser alterados por um algoritmo interno. Somente o conjunto de dados é mantido na memória para uma determinada instância do bloco funcional.

Blocos funcionais podem ser utilizados para a criação de outros blocos funcionais (blocos derivados), aumentando ainda mais a capacidade de reutilização do software.

Funções

Funções são elementos de software que não aparecem no modelo de software. Funções não possuem persistência, existindo apenas em tempo de execução, assim como sub-rotinas. Portanto, sempre produzem o mesmo resultado para o mesmo conjunto de entradas.

Funções têm apenas uma saída, sem considerar a saída ENO para controle de execução, ao contrário dos Blocos de Funções que podem ter várias. O resultado

pode ser um tipo de dado simples de múltiplos elementos (vetores e estruturas). As funções trigonométricas são os tipos mais comuns de funções.

Reutilização de Programas, Blocos Funcionais e Funções

Pela norma, programas, blocos funcionais e funções são considerados Unidades de Organização de Programas (POU). A principal finalidade destes elementos é a possibilidade de reutilização através de instâncias (exceto as funções, que não são persistentes). Desta forma, a reutilização pode ser em macro, por programas, ou em micro escala por blocos funcionais. A recursividade não é permitida por norma dentro de uma POU por motivos de estabilidade e de segurança da aplicação.

A utilização de *Function Blocks* e *Functions* padrões, dá-se através de bibliotecas fornecidas pelo fabricante do CLP, ou pela criação de blocos e funções específicos definidos pelo usuário para atender às necessidades do processo.

Tipo de POU	Aplicado como	Comentário
Programa	Instância de um Programa	Permite reutilização no nível macro, como programas para reatores, transportadores, caldeiras, etc.
Bloco Funcional	Instância de um Bloco Funcional	Possibilita a reutilização desde simples a complexas estratégias de controle e algoritmos, como controle PID, filtros, motores, etc.
Função	Função	Usada para tratamento comum de dados, como lógica E, OU, seno, cosseno, soma e etc.

Tabela 5.10. Exemplos de Reutilização de POUs

Tarefas (Tasks)

Uma *Task* é um mecanismo de escalonamento muito útil para sistemas de tempo real, que executa Programas ou Blocos funcionais periodicamente ou em resposta a um evento (mudança de estado de alguma variável booleana), permitindo a execução de programas em diferentes taxas.

A necessidade de se executar programas em taxas diferentes tem por objetivo atender as exigências de tempo de resposta do processo controlado e de otimizar o uso da capacidade de processamento do CLP.

Um forno, por exemplo, que possui uma capacidade térmica muito grande, pode ser controlado por um algoritmo que executa uma vez a cada minuto, permanecendo estável. Já as funções de intertravamento de segurança de uma máquina ferramenta devem ser executadas, por exemplo, a cada 5 ms. A cada tarefa podemos atribuir um período de execução e uma prioridade.

A norma IEC assume que tarefas em diferentes recursos sempre executam de forma independente. Entretanto, em algumas implementações, pode ser necessário a utilização de mecanismos de sincronização.

Neste contexto, uma simples execução de um Programa ou Blocos funcionais implica que todos os elementos de software dentro dos mesmos são processados uma vez.

A norma IEC não define nenhum mecanismo implícito para execução de Programas. Ou seja, um Programa ou Blocos funcionais ficará aguardando a sua execução até que seja associado a uma determinada Tarefa e esta seja ativada por uma execução periódica ou por um determinado evento (trigger).

Um Bloco funcional devidamente associado a uma tarefa, será executado na mesma Tarefa do Programa onde esteja contido (programa pai).

Existem 2 tipos diferentes de *Task*: Preemptivas e Não-Preenptivas, sendo estas orientadas a eventos (trigger) ou a tempo (periódicas).

Quando existem tarefas múltiplas, normalmente são atribuídos diferentes intervalos e prioridades para cada uma. Para permitir o funcionamento das tarefas, podem ser utilizados dois métodos de escalonamento, o preemptivo e o não-preemptivo. O método adotado por um CLP pode alterar significantemente o comportamento do sistema.

Task não-preemptiva

Uma tarefa não-preemptiva sempre completa seu processamento, uma vez iniciado, se sofrer interrupções. O intervalo entre a execução de *Tasks* pode variar muito neste tipo de escalonamento.

Task preemptiva

É recomendado para sistemas que devam apresentar comportamento determinístico no tempo. Neste sistema quando o intervalo de uma *Task* de maior prioridade vence, a *Task* em execução sofre preempção (é suspensa) e a nova *Task* de prioridade maior passa a executar imediatamente. Quando a *Task* de maior prioridade termina, a *Task* suspensa anteriormente volta a executar do ponto onde parou.

Variáveis de escopo Local e Global

A norma exige a declaração de variáveis dentro de diferentes elementos de software, tais como Programas e Blocos Funcionais. As variáveis podem utilizar nomes com significado abrangente (simbólicos) e serem de diferentes tipos de dados definidos por norma.

Podem ser de alocação dinâmica ou associadas a posições de memória (representação direta). O escopo das variáveis é local ao elemento de software que as declara, permitindo acesso dentro do próprio elemento que pode ser uma Configuração, Recurso, Programa, Bloco Funcional ou Função. Variáveis também podem ser de escopo global, sendo acessadas por todos os elementos contidos no mesmo, incluindo os elementos aninhados.

A seguir um exemplo da utilização de variáveis globais, onde a variável EFETIVO_TEMP é utilizada no Programa_1 (*Program*) e no FB_1 (*Function Block*) contido no Programa_2 (*Program*):

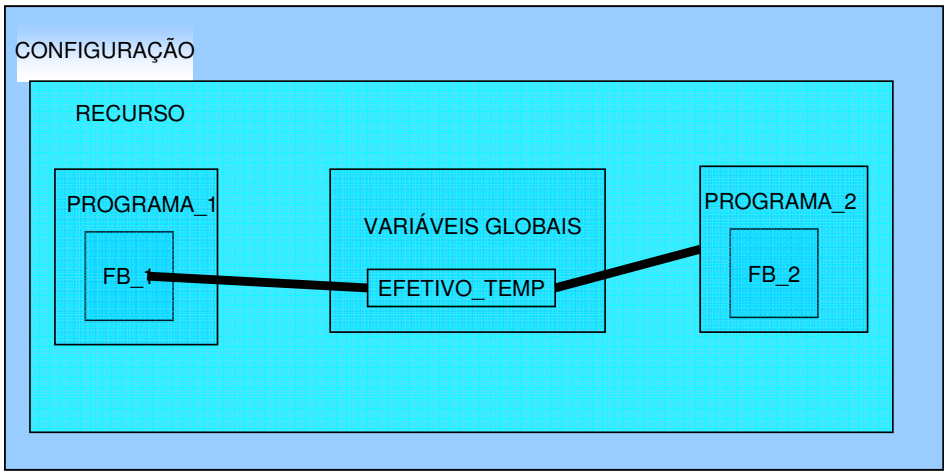


Figura 5.11. Variáveis Globais

Variáveis de representação direta

Posições de memória do CLP podem ser acessadas usando variáveis de representação direta. A representação direta permite a leitura e escrita de dados em posições conhecidas de memória, tais como entradas, saídas e endereços internos. As variáveis de representação direta têm seu uso restrito aos Programas não permitindo que Blocos Funcionais façam acesso direto para garantir a reutilização dos blocos. A notação utilizada é padronizada para permitir a portabilidade.

Todas começam com o caracter % seguido de uma ou duas letras.

Sinal inicial (IEC std)	**Identificação de memória**	**Tamanho do dado**		**DESCRIÇÃO**
%	**M** (Acesso à memória)	**X**	(1 bit)	Acesso à variáveis booleanas.
		W	(16 bits)	Acesso à variáveis com 16 bits de tamanho: INT, UINT e WORD.
	I (Entrada física do CLP)	**D**	(32 bits)	Acesso à variáveis com 32 bits de tamanho: DINT, UDINT, DWORD,
		T	(32 bits)	TIME, DATE, TOD e DATE_AND_TIME.
	Q	**R**	(32 bits)	Acesso à variáveis com 32 bits de tamanho: REAL

	(Saída física do CLP)	**A**	(8 bits)	O conteúdo dessa região é definido pelo usuário conforme a necessidade do projeto.
Exemplos de acesso a dados utilizando a nomenclatura IEC61131-3:				
%I0.0	Leitura de sinal da entrada E0 da CPU (slot 0, entrada 0)			
%IW8.0	Leitura de sinal da entrada proveniente de uma placa analógica (slot 8, entrada 0)			
%Q0.0	Escrita de sinal na saída S0 da CPU (slot 0, saída 0)			
%QW8.0	Escrita de sinal na saída proveniente de uma placa analógica (slot 8, saída 0)			
%MX1	Acesso à leitura/escrita de variável booleana na memória do CLP			
%MW1	Acesso à leitura/escrita de variável de 16 bits na memória do CLP			
%MD1	Acesso à leitura/escrita de variável de 32 bits na memória do CLP			
%MT1				
%MR1				
%MA1	Acesso à leitura/escrita de variável de 8 bits na memória do CLP			

Tabela 5.12. Sintaxe de Representação Direta de Variáveis

Atributos de variáveis:

- Retenção: Indica que as variáveis seguintes serão colocadas em memória retentiva, isto é que mantêm o seu valor em caso de perda de alimentação do CLP. São declaradas utilizando Retain.
- Constante: Indica que os valores de uma lista de variáveis não podem ser modificados.

Caminho de Acesso (Access Paths):

Os caminhos de acesso permitem a transferência de dados entre diferentes configurações. Cada configuração pode definir um número de variáveis para acesso por configurações remotas. A norma assume que estarão disponíveis mecanismos de comunicação para troca de informações, não abordando a forma a ser adotada.

Fluxo de controle

A norma IEC não define os mecanismos para controle de execução dos elementos de software, os quais são dependentes da implementação. Entretanto, são definidos os comportamentos na partida e parada do sistema:

Partida:

Quando uma configuração parte, todas as variáveis globais são inicializadas e todos os recursos são atividados.

Quando um recurso parte, todas as variáveis dentro do recurso são inicializadas e todas tarefas são habilitadas.

Uma vez habilitada as tarefas, todos os programas e blocos funcionais associados às mesmas executarão quando a tarefa estiver ativa.

Parada:

Quando uma configuração para, todos os recursos da mesma param.

Quando um recurso para, todas as tarefas são desabilitadas, interrompendo a execução dos programas e blocos funcionais.

Deve ser observado que um programa somente controla a execução dos blocos funcionais associados à mesma tarefa. Entretanto, os blocos funcionais podem ser associados a tarefas distintas, não sendo necessariamente sincronizados com os programas.

Exemplos de sistemas reais aplicando o modelo de software IEC

Em aplicações com somente uma CPU, o modelo a ser adotado será constituído por uma configuração e um recurso, podendo ter um ou mais programas. Conforme figura 7:

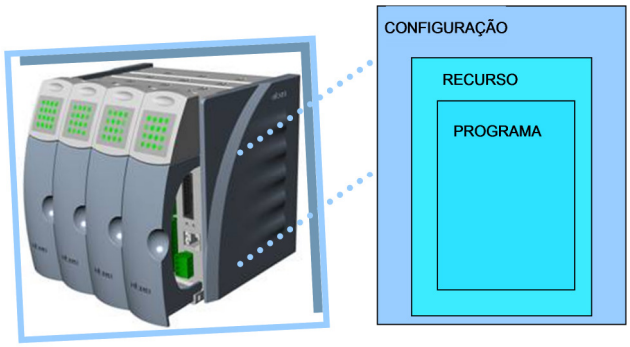


Figura 5.13. Configuração com um único CLP

Em aplicações de grande porte com diversos CLP's, pode-se considerar uma única configuração que se refere à planta, com diversos recursos, onde cada CLP corresponderá a um recurso. Conforme figura 8:

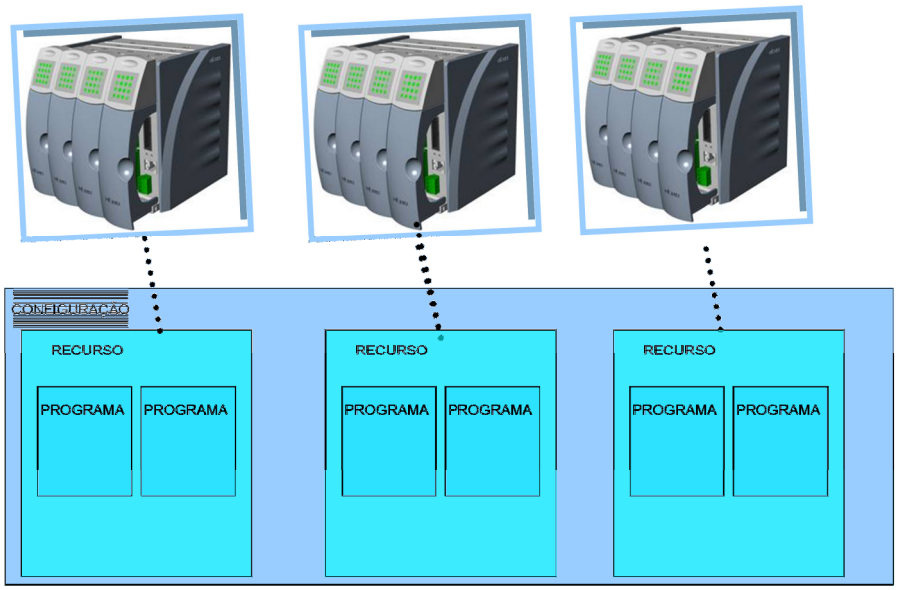


Figura 5.14. Configuração com diversos CLP's

Aplicações com mais de uma planta ou áreas interligadas, com o uso de diversos CLP's podem ser considerados como uma ou várias configurações. Conforme figura 9:

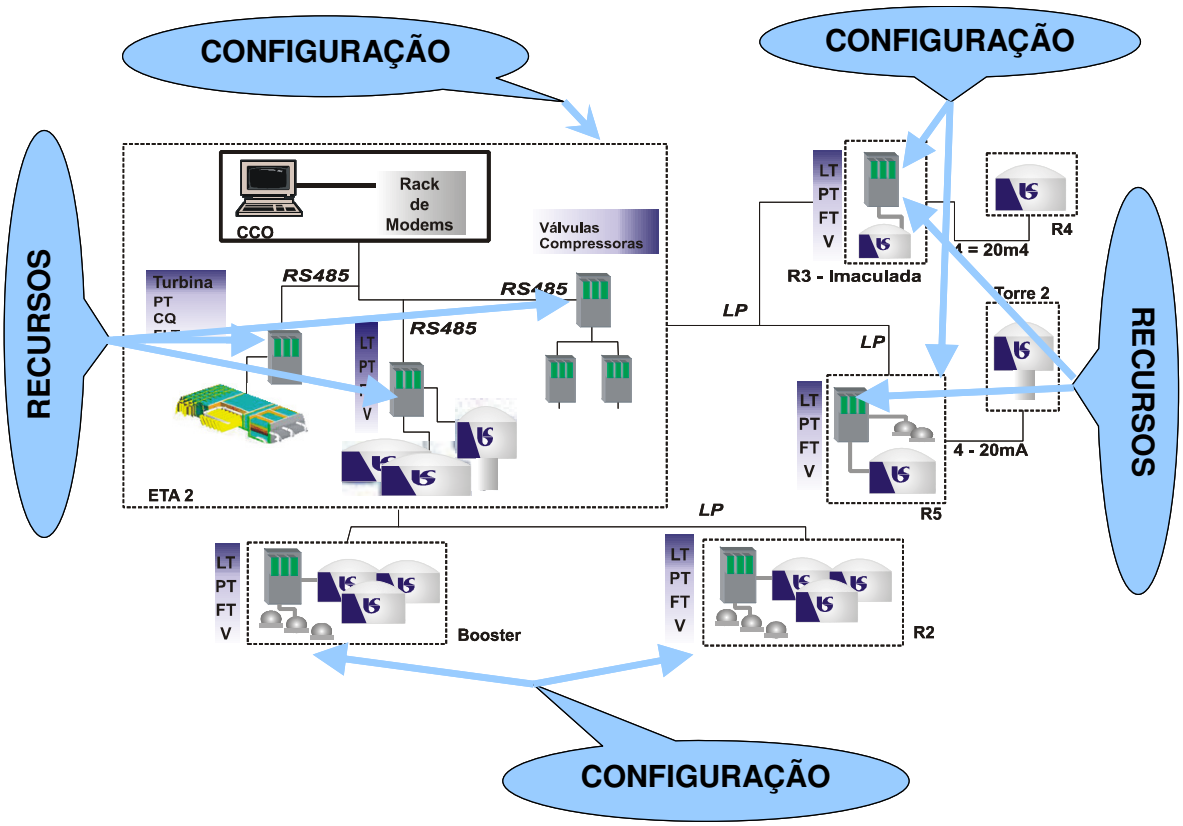


Figura 5.15. Configuração com diversas estações.