SUPORTE DE CURSO

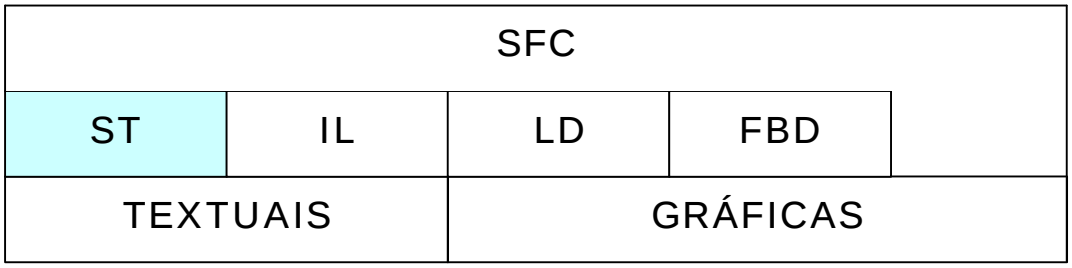
IEC 1131-3 Texto Estruturado

Livro Texto: Programming industrial control systems using IEC 1131-3 – R.W. Lewis

UFMG – Informática Industrial
Prof. Constantino Seixas Filho

IEC 1131-3 Texto Estruturado

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



É usada para descrever o comportamento de :

- Funções
- Blocos de funções
- Programas
- Em SFC para expressar o comportamento de passos, ações e transições.

Atribuição

X:=Y;

Exemplo:

```
TYPE Alarm
   STRUCT
      TimeOn: DATE_AND_TIME;
      Dutration: TIME;
   END_STRUCT;
END_TYPE

VAR
      Rate, A1: REAL;
      Count: INT;
      Alarm: Alarm;
      Alarm2: Alarm;
      Profile: ARRAY[1..100] OF REAL;
```

```
        RTC1: RTC;      (* Real time clock *)
END_VAR;

Rate:= 13.1;            (* Valor literal i.e. constant *)
Count := Count +1;      (* Expressão simples *)
A1 := LOG(Rate);        (* Valor da função *)
Alarm1.TimeOn := RTC1.CDT; (* Parâmetro de saída de um bloco de função *)
Alarm2 := Alarm1;       (* Variável Multi elemento *)
(* Valor de uma expressão complexa é atribuído a único elemento de um array *)
Profile[3] := 10.3 + SQRT((Rate + 2.0) * (A1/2.3));
```

As operações de atribuição devem ser atômicas.

Expressões:

Expressões geram valores derivados de outras variáveis e constantes. Expressões devem produzir valores do mesmo tipo de uma variável sendo atribuída.

Operadores:

Operador	Descrição	Precedência
(…)	Expressão parentisada	
Função(…)	Lista de parâmetros de uma função	
**	Exponenciação	
- NOT	Negação Complemento booleano	
* / MOD	Multiplicação Divisão Operador de módulo	
+ -	Soma Subtração	
<,>,<=,>=	Comparação	
= <>	Igualdade Desigualdade	
AND, &	E booleano	
XOR	OU Exclusivo booleano	
OR	OU booleano	

Exemplo:

```
Area:= PI * R * R;
V := K** (-W * T);          (* K elevado à potência -W * T)
Volts := Amps * Ohms;
```

Status := (Valve1 = Open) XOR (Valv1 = Shut);

Avaliação de expressões:

Quando os operadores têm a mesma precedência, eles são avaliados esquerda para a direita.

Existem funções equivalentes aos operadores do ST.

	50.0
Velocidade2	
Pressão	30.0

Taxa := Velocidade1/10.0 + Velocidade2/20.0 SQRT(Pressão + 6.0);

Ordem de avaliação:

Velocidade1/10.0 = 5.0

Pressão + 6.0 = 6.0
=8.0 (Velocidade1/10.0 + Velocidade2/20.0)
8.0 - (8 SQRT(Pressão + 6.0))

0.0 + Velocidade2/(20.0 -

- 6.0)
Rate = 5 + 60/14.0

deduzido.

StartUp := A AND B AND D AND E;

Se A = FALSE então o valor FALSE é atribuído à variável StartUp e a

Chamada de blocos de função

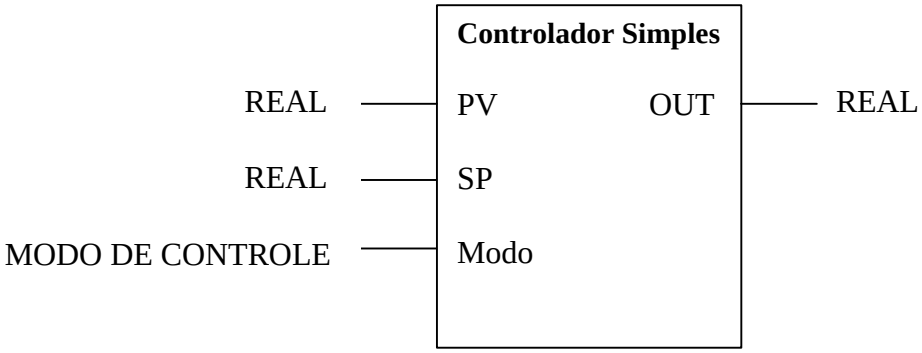
Podemos invocar um bloco de função através de seu nome, tendo como parâmetros valores válidos para as parâmetros de entrada.

Forma Geral:

InstânciaBlocoFunc(ParamEntrada1:=Expressão1, ParamEntrada2:=Expressão2, …);

Se algumas entradas ficarem indefinidas na chamada do bloco de função, e o bloco estiver sendo chamado pela primeira vez, o valor default do parâmetro será assumido. Se o bloco já tiver sido chamado anteriormente o último valor da entrada será assumido.

Exemplo:



Loop1 é uma instância do bloco de funções SimpleControl, que corresponde a um controlador PID. Os modos de operação são: AUTO e MANUAL.

(* Primeira invocação *)
Loop1(PV:= Input1.Out + Offset, SP := 100.0);

O valor default para Modo é assumido como AUTO.

(* Invocação Posterior *)
Loop1(PV:= Input.Out + Offset, Modo := MANUAL);

SP não é definido.
O último valor utilizado é assumido: 100.00.

Seria desejável definir valores para os parâmetros de entrada dos blocos de função antes da invocação do bloco.
Loop1.Mode := MANUAL;
Esta construção NÃO é definida no padrão e portanto NÃO é PORTÁVEL.

Os valores de saída de instâncias de blocos de função são sempre disponíveis:

Exemplo:

Vazão := Loop1.Out;

Comando condicional

IF <expressão booleana> **THEN**
    <comandos>
END_IF;

IF <expressão booleana> **THEN**
    <comandos>
ELSE
    <comandos>
END_IF;

IF <expressão booleana> **THEN**
    <comandos>
ELSIF <expressão booleana>
    <comandos>
ELSE
    <comandos>
END_IF;

Exemplos:

IF Colisão **THEN**
    Velocidade := 0;
    Freios := ON;
END_IF;

IF (Porta = FECHADO) AND (Bomba = ON) AND (Temp > 200.0) **THEN**
    EstadoControle := Ativo;
ELSE
    EstadoControle := HOLD;
    VelocidadeBomba := 10.0;
END_IF;

```
IF                  THEN
        IF TamanhoChama > 4.0

        ELSE

        END_IF
ELSE
        Fuel := 1000.0;
END_IF;


    A> B THEN
        D:= 1;
ELSEIF A= B + 2

ELSEIF      - 3

ELSE

END_IF
```

CASE

```
CASE <expressão inteira> OF
        < valor de seletor inteiro> : < comandos>
        < valor de seletor inteiro> : < comandos>
        < valor de seletor inteiro> : < comandos>
ELSE
        <comandos>
END_CASE;
```

Exemplo

```
CASE seletor_velocidade OF
1:      velocidade := 10.0;
2:      velocidade := 20.4;
3:      velocidade := 30.0;     Ventilador1 := ON;
4,5:    velocidade := 50;       Ventilador2 := ON;
6..10:  velocidade := 60.0;     Agua := ON;
ELSE
        velocidade :=0; FalhaVelocidade := TRUE;
END_CASE;
```

Comandos de repetição:

FOR..DO

FOR <inicializa variável de iteração >
TO < valor final da expressão>
BY < expressão incremental> **DO**
    <comandos>
END_FOR;

- A expressão precedida por **BY** é opcional. Se **BY** for omitido, a variável de iteração será incrementada de 1.
- A variável de iteração NÃO deve ser utilizada fora do loop, pois seu valor é dependente da implementação.
- As instruções dentro do comando não devem modificar as variáveis que afetam os valores de incremento e final.

Exemplo

```
FOR I:=100 TO 1 BY -1 DO
    Canal[I].status := ON;
END_FOR;

FOR T:= NumTanques -1 TO TanqueMax * 2 DO
    NumTanque := T;  volume(tanque:= T);
END_FOR
```

WHILE..DO

WHILE <Expressão booleana> **DO**
    <comandos…>
END_WHILE

Exemplo

```
WHILE Value < (ValorMax -10.0) DO
    MovePonte();
    Valor := Valor + Ponte.Posicao;
END_WHILE
```

REPEAT..UNTIL

```
REPEAT
        <comandos>
UNTIL <expressão booleana>
END_REPEAT
```

Exemplo

```
Tentativas := 0;
REPEAT
        Tentativas: =  Tentativas + 1;
        CambioMotor1(Modo :=  DESABILITADO);
UNTIL (CambioMotor1.Estado = OFF) OR (tentativas > 4)
END_REPEAT
```

Exit

Só pode ser usada dentro de iterações.
Aborta loop.

Exemplo

```
Falta := FALSE;
FOR I:= 1 TO 20 DO
        FOR J:= 0 TO 9 DO
                IF ListaEquipamentosFalta[I, J] THEN
                        NumeroFalta := I*10 + J;
                        Falta := TRUE; EXIT;
                END_IF;
        END_FOR;
        IF Falta THEN EXIT;
END_FOR;
```

Return

Só pode ser usado dentro de funções e blocos de função.
Aborta a execução da função ou bloco de função.

Exemplo

```
FUNCTION_BLOCK TEST_POWER
    VAR_INPUT
        Corrente, Tensao1, Tensao2, Tensao3: REAL;
    END_VAR
    VAR_OUTPUT
        Sobretensao: BOOL;
    END_VAR

    IF Tensao1 * Corrente > 100 THEN
        Sobretensao := TRUE; RETURN;
    END_IF;
    IF Tensao2*(Corrente+10.0) > 100 THEN
        Sobretensão := TRUE; RETURN;
    END_IF;
    IF Tensao3*(Corrente+20.0) > 100 THEN
        Sobretensao := TRUE;
    END_IF;
END_FUNCTION_BLOCK;
```

Dependências de implementação:

Algumas limitações de ordem práticas podem existir em algumas implementações tais como:

- Tamanho de expressões.
- Tamanho de comandos
- Tamanho de comentários
- Número de opções em instruções **CASE**
- Etc

Exemplos em texto estruturado

Resolvendo a equação do segundo grau

```
Delta=  b*b - 4*a*c;
IF Delta > 0.0 THEN
    RaizDelta : = SQRT(Delta);
    X_Root1 := (-b + RaizDelta)/2*a;
    X_Root2:= (-b - RaizDelta)/2*a;
ELSE
    msg := 'Raízes Imaginárias';
END_IF;
```

Calculando a Média, o Máximo e o Mínimo

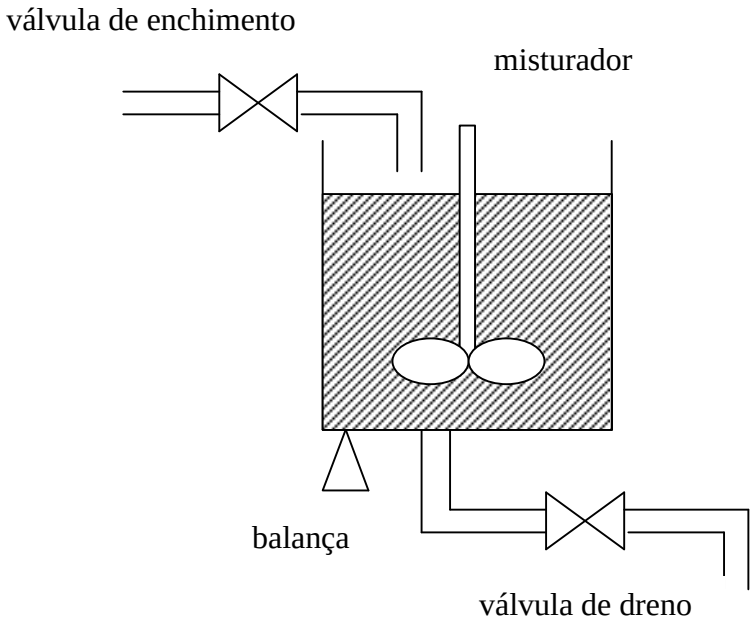
```
TYPE T_Channel:
        STRUCT
                value: REAL;  state: BOOL;
        END_STRUCT;
END_TYPE

TYPE
         T_Inputs: ARRAY[1..32] OF T_Channel;
END_TYPE

VAR
        max: REAL := 0;          (* inicializa com menor valor possível *)
        min: REAL := 10000.0;(* inicializa com maior valor possível *)
        input : T_Inputs AT % IW130;          (* canais de entrada *)
        soma: LREAL := 0.0;                   (* somatório dos valores *)
        media : LREAL;
END_VAR

FOR I:= 1 TO 32 DO
        soma:= REAL_TOLREAL(input[I].value) + soma;
        IF input[I].value > max THEN
          max := input[I].value;
        END_IF;
        IF input[I].value < min THEN
          min := input[I].value;
        END_IF;
END_FOR;
media : = soma / 32.00;
```

Bloco de controle de um tanque




```
(* Estado do tanque *)
TYPE T_STATE: (CHEIO, NAO_CHEIO, VAZIO); END_TYPE;
(* Estado da válvula *)
TYPE T_VALVE: (ABERTA, FECHADA); END_TYPE;

FUNCTION_BLOCK TankControl
   VAR_IN  (* Parâmetros de entrada *)
      Comando: SINT;
      Peso: REAL;
      PesoCheio, PesoVazio: REAL;
   END_VAR
```

```
VAR_OUTPUT
   Enche:        T_VALVE := FECHADA;
   Esvazia:      T_VALVE := FECHADA;
   Velocidade:   REAL := 0.0;
END_VAR

VAR
    Estado: T_STATE := VAZIO;
END_VAR

(* Corpo do bloco de função *)

(* Determina o estado do tanque *)
IF Peso >= PesoCheio THEN          (* Está cheio ? *)
    Estado:= CHEIO;
ELSIF Peso <= PesoVazio THEN       (* Está Vazio ? *)
    Estado := VAZIO;
ELSE
    Estado := NAO_VAZIO;
END_IF;

(* Processa modo de comando *)
CASE Comando OF
1:  ValvDreno := FECHADA;          (* Enche tanque *)
    ValvEnche:= SEL(G:= Estado, IN0:=ABERTA, IN1:= FECHADA);
2:  ValvDreno := FECHADA;          (* Mantém conteúdo *)
    ValvEnche := FECHADA;
4:  ValvDreno := ABERTA;           (* Esvazia Tanque *)
    ValvEnche := FECHADA;
END_CASE;

(* Controla a velocidade do motor do agitador *)
Velocidade := SEL(G:= ((Comando= 3) AND (Estado = CHEIO));
                        IN0:=0.0; IN1:= 100.0);
END_FUNCTION_BLOCK
```

Observações:
Em uma aplicação real

- Um bloco de funções como este ficaria associado a uma task e o algoritmo seria reavaliado a cada x ms.
- Os valores de entrada devem ser verificados quanto a filtragem de ruído, faixa normal de operação, etc.

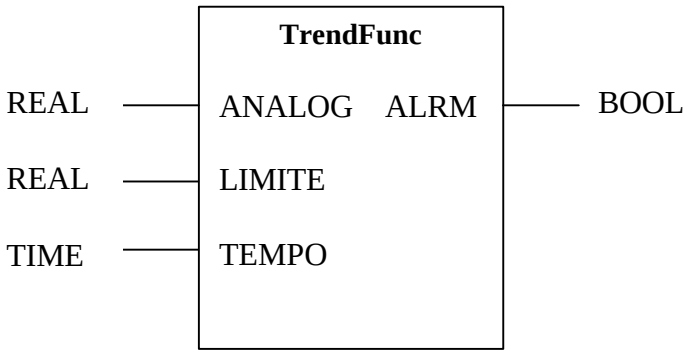
Leitura Complementar:

- Bonfatti, Monari, Sampieri, IEC1131-3 Programming Methodology, CJ International, 1997.

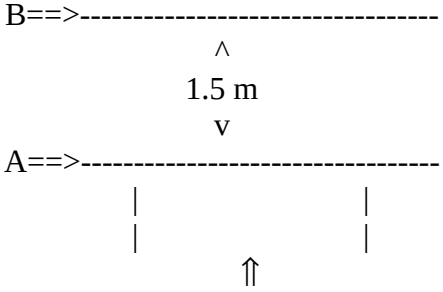
Exercícios:

1. Escreva um bloco para totalizar uma variável do tipo vazão. Uma das entradas corresponde à vazão instantânea em unidade/hora e a outra a um pulso de sincronismo de 0.1 segundo. O bloco deve dar o resultado acumulado, que pode ser amostrado a qualquer instante.

2. Escreva um bloco que examina o estado de uma variável no tempo. Se o valor da variável tender a um limite máximo estipulado, num tempo x, uma saída de alarme deve ser setada.



3. Em um estacionamento duas células fotoelétricas distanciadas de 1.5 metros servem para identificar a entrada e saída de carros. Quando a célula A é acionada primeiro que a célula B é porque um veículo está entrando. No caso contrário um veículo está saindo. Se após uma das células ter seu facho de luz bloqueado, a outra não for interrompida, isto significa que uma pessoa está passando e o evento deve ser ignorado. A variável cont indica o número de veículos no estacionamento. Desenvolva um bloco que tenha como entrada a leitura dos sensores fotoelétricos (booleanos) e como saída a contagem de carros na garagem, o número de vagas disponíveis, e a sinalização de um semáforo vermelho caso o número de vagas seja 0.



4. Escreva um programa que dê o volume instantâneo de um líquido em um tanque cilíndrico, dados a altura máxima do tanque, o diâmetro da base e o peso do líquido no cilindro e a temperatura. A densidade do líquido em CATP é um parâmetro de entrada, assim como o coeficiente de expansão volumétrica do líquido.

5. Desenvolva um bloco para cálculo do CRC16 dado um array de N bytes.

6. Faça um programa para inverter um string dado:

```
VAR
    Nome: STRING(13) = 'ORA_PRO_NOBIS';
    Cont: INT=0;
    NomeRev: STRING(13) = '';
END_VAR
```