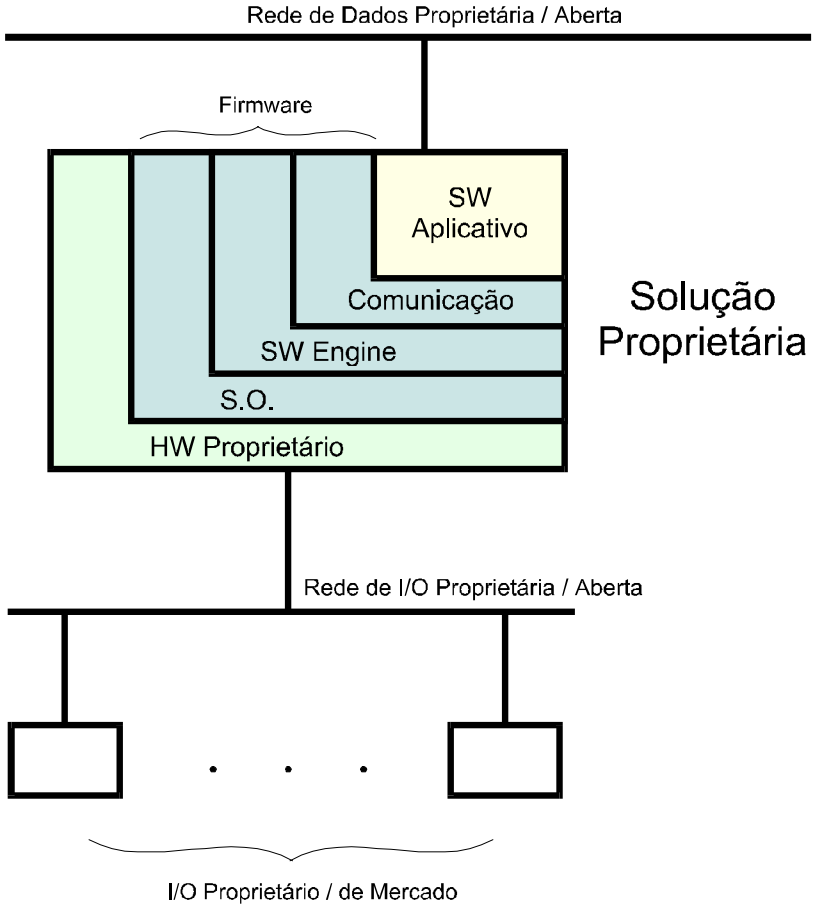
ATAN

CPDEE

Aula
IEC 61131-3

dezembro de 1999

O que é um CLP ?



É uma solução proprietária de HW e SW para aquisição de dados e controle de processos

Características básicas

- Sistema modular
- Hardware dedicado (CPU, memória, I/O, comunicação, etc.)
- Software fechado (S.O., comunicação, etc.)
- Sistema de tempo real
- Execução cíclica do aplicativo (CLP padrão)
- Limitação da capacidade e dos recursos de HW e SW por classe de produto
- Software aplicativo dependente do hardware
- Alta integração entre HW e SW aplicativo

Vantagens

- Grande aceitação do mercado
- Facilidade de configuração / programação
- Solução robusta de HW e SW
- Alta confiabilidade (Redundância, *Fail Safe*, Troca a quente, etc.)
- Desempenho comprovado
- Tecnologia sedimentada

Desvantagens

- Solução proprietária de HW e SW
- Custo elevado dos módulos inteligentes
- Custo elevado para expansão
- Requer maior especialização / treinamento no produto
- Dificuldade de atualização e expansão de HW
- Capacidade de processamento restrita em comparação aos sistemas computacionais

Visão do Usuário (OMAC, 1994)

Dificuldades encontradas nos sistemas atuais

- Tecnologia proprietária
- Política comercial (ditadura de preços)
- Interfaces proprietárias
- Elevado custo para integração
- Treinamento orientado ao produto
- Alto custo para expansão

Necessidades (*CNC*, *Motion*, *Discret Control*)

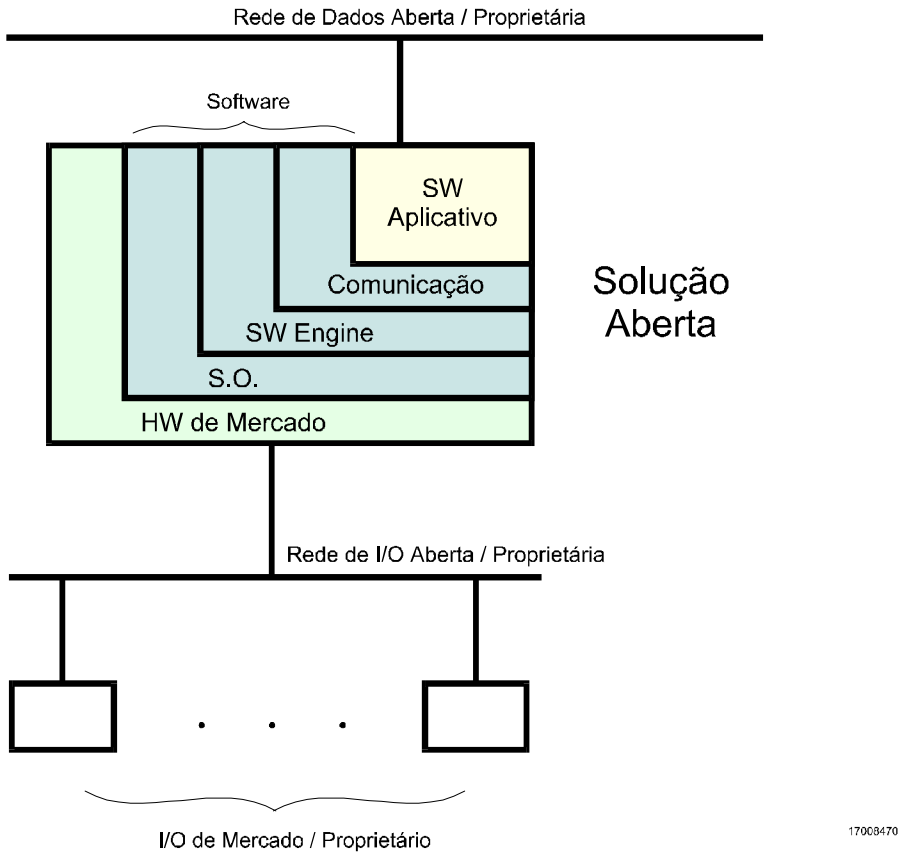
- **Segurança, confiabilidade e robustez**
- **Baixo custo**
- **Flexibilidade**
- **Conectividade**
- **Mantenabilidade**
- **Facilidade de treinamento**

Desejo do Usuário (OMAC)

Open Modular Architecture Controller **implica em uma solução**

- **Econômica**
- **Aberta**
- **Modular**
- **Fácil manutenção**
- **Escalável**

O que é *Soft Logic*?



É uma solução de SW baseada em HW aberto para aquisição de dados, controle de processos e ...

Características

- Hardware dedicado ou de mercado (padrão PC)
- Software aberto (S.O., comunicação, etc.)
- Sistema de tempo real dependente da combinação de HW e SW
- Execução periódica ou engatilhada (multitarefa)
- Recursos de HW ilimitados
- Software independente do hardware
- Integração entre HW e SW aplicativo através do S.O.

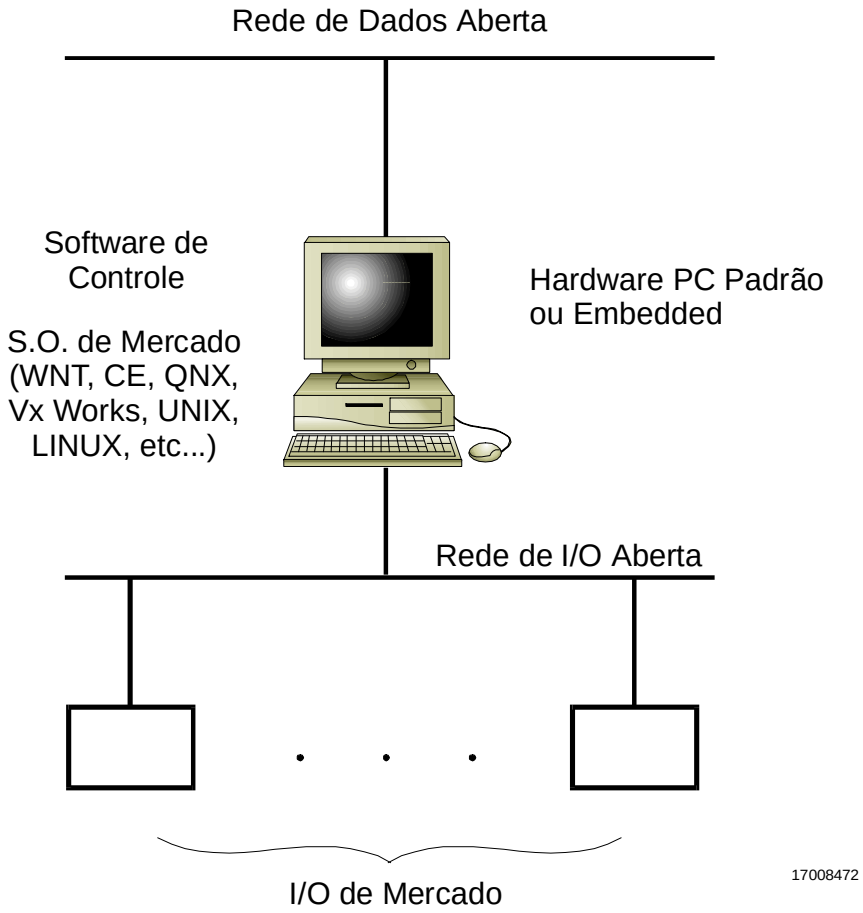
Vantagens

- Solução aberta
- Baixo custo de HW (HW de mercado)
- Maior desempenho (HW de última geração)
- Maior integração entre sistemas (controle, supervisão, BD, MES, ERP, etc.)
- Maior funcionalidade (CLP, CNC, *Motion Control*, DCS, IHM, etc.)
- Flexibilidade
- Facilidade de expansão
- Maior conectividade
- Facilidade de manutenção / treinamento
- Etc...

Desvantagens

- **Baixa aceitação devido ao desconhecimento e falta de experiência do mercado**
- **Confiabilidade questionável em aplicações críticas**

O que é controle baseado em PC?



É uma solução Soft Logic baseada na arquitetura PC

Em 1992, o IEC publicou a norma IEC 61131, a qual estabelece padrões para Controladores Programáveis

A norma 61131 é dividida em 5 partes

- **61131-1 - Informações gerais**
- **61131-2 - Requisitos de hardware**
- **61131-3 - Linguagens de programação**
- **61131-4 - Guia de orientação ao usuário**
- **61131-5 - Comunicação**

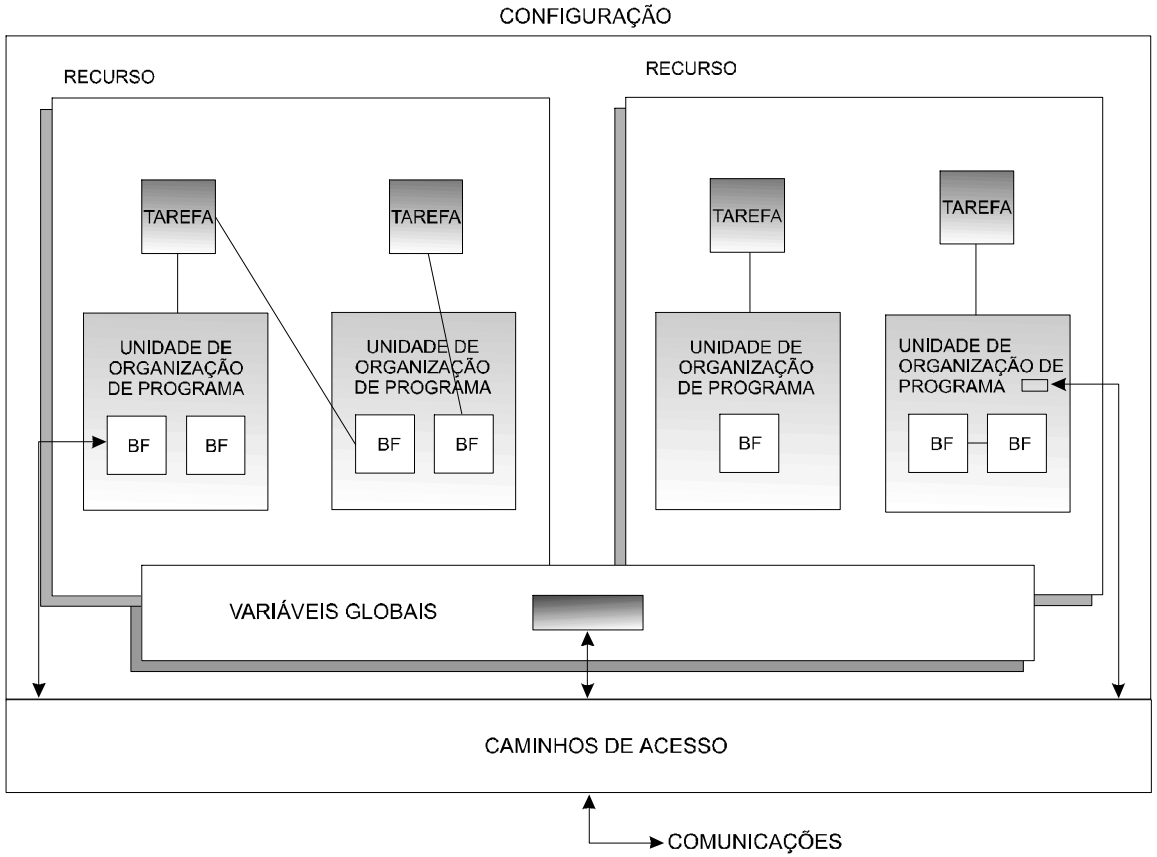
Outras três partes estão em fase de elaboração

- **61131-6 - Comunicação via *Fieldbus***
- **61131-7 - Programação utilizando Lógica *Fuzzy***
- **61131-8 - Guia para implementação das linguagens**

Norma IEC 61131

Combina linguagens de SDCD e CLP numa solução única

Objetivo: tornar SW de CLP modular, reutilizável e portável



Modelo de Estruturação do Software

Definições

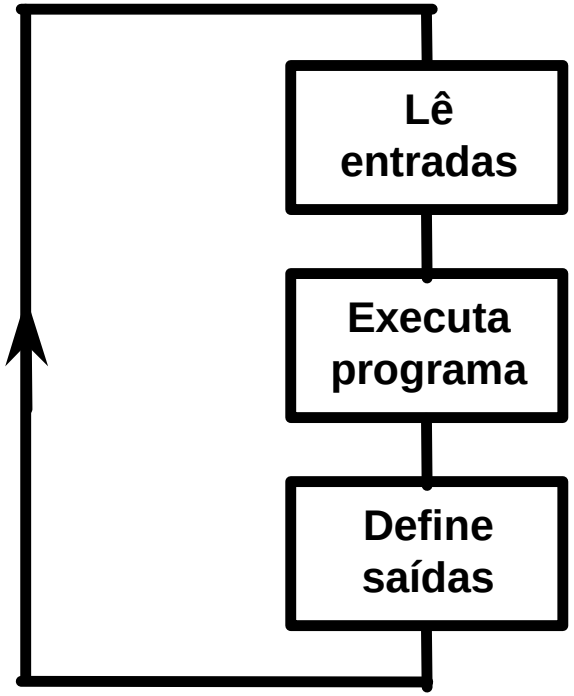
- **Configuração**
  Corresponde ao sistema do controlador programável, composto pelos diversos recursos
- **Recurso**
  Corresponde a uma função para processamento de sinais e suas funções para Interface Homem-Máquina (IHM) e interface com sensores e atuadores
- **Tarefa**
  Elemento para controle de execução periódica ou engatilhada (condições de disparo = eventos) de um grupo de unidades
- **Unidade de organização de programa (POU)**
  Uma função, bloco de função ou programa

Definições

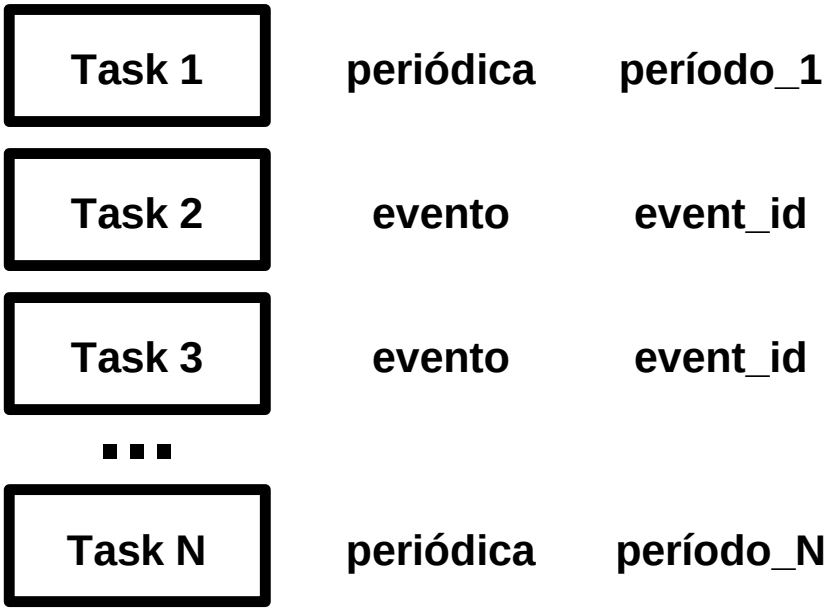
- **Variável global**
  Variável cujo escopo é global
- **Caminho de acesso**
  Associação de um nome simbólico para a variável a fim de associar as entradas/saídas físicas ao SW de processamento de variável
- **Endereçamento hierárquico**
  Representação direta de um elemento de dado como membro de uma hierarquia física ou lógica
- **Tipos de dados**
  Conjunto de valores + conjunto de ações permitidas (procedimentos)

Mecanismo de execução

CLP tradicional



CLP IEC 61131



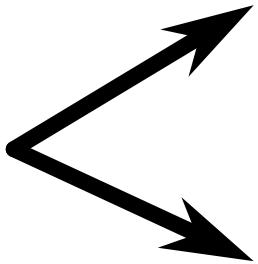
- Ambiente multitarefa
- Usuários devem definir cada variável antes de usá-la

Norma IEC 61131

"Apesar do esforço estabelecido pela norma IEC 61131-3 para padronização das linguagens de programação, alguns aspectos da norma são dependentes da forma de implementação"

"Empresas americanas não se preocupam muito com normas, elas possuem poucos engenheiros, escolhem um fornecedor e ficam ligadas a ele"

Open PLC



Controle baseado em PC

Soft PLC

Programação transportável (IEC 61131-3)
PLCopen - Netherlands - 1992

"Opennes implies more than open communication networks, it also implies open software" **[The state of PLC - InTech - April - 96]**

Open PLC

PLC Open é uma associação independente que promove uma padronização na implementação da norma IEC 61131-3

- **Atuação**
  - **Divulgação da norma**
  - **Definição de interfaces comuns**
  - **Certificação de produtos**
- **Testes de compatibilidade**
  - ***Base level***
  - ***Portability level***
  - ***Full compliance***
- **Recomendações sobre utilização da norma**

Características dos CLPs abertos baseados em PC

- Maior velocidade que CLPs convencionais
- Não há limite de quantidade de memória
- Maiores facilidades de comunicação via diversas tecnologias de rede
- Múltiplas linguagens de programação (padrão 61131-3)
- Incorpora todas as vantagens e funcionalidades dos PCs convencionais

Mercado para o CLP aberto hoje

- Automobilística
- Bebidas
- Alimentos
- Cigarros
- Farmacêutica
- Predial
- Manufatura em geral

Produtos *Soft Logic* (*Control Engineering* - Março, 1999)

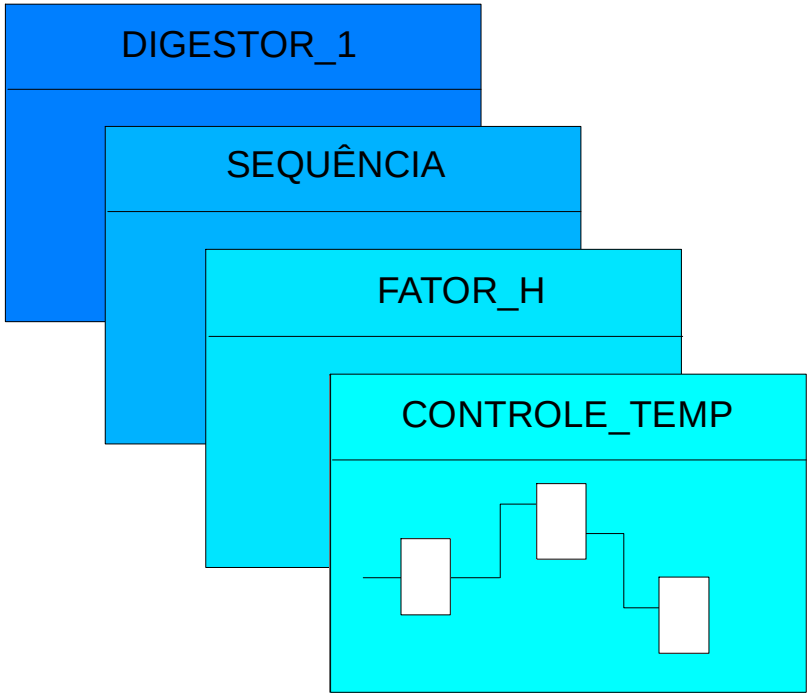
PC Control Vendors		
Company	**Software**	**Strategy**
3S	CoDeSys	Windows NT
ABB	Advant OCS	Windows NT, RTX
Adept Technology	AdeptWindows	NT, 95
Aerotech Inc.	U500, U600	95, NT
Afcon	P-CIM	NT
AlterSys	Soft logic/DCS, solution PCP Virgo	QNX
Apilogic	Apigraf	NT
ASAP	ASIC-300	NT
Aspen Technology	Plantelligence	NT
Cegelec	Alspa P3200 for utilities	Java
Ci Technologies	Citect	NT
CJ International	IsaGraf	NT
Comdale Technologies	Expert systems	QNX
Control Systems International	Field Control System	QNX
CTC Parker	MachineLogic	DOS/RTX
Cutler-Hammer/Eaton	NetSolver	NT/INtime
Cyberonics	HMI & PLCs for process control	Java
Elsag Bailey	Symphony, Freelance 2000	NT, NT
Fisher-Rosemount	PlantWeb	NT
Foxboro	FoxDMM	Java
GE Fanuc	PC Control	NT
Hewlett-Packard	Chai	Java
Honeywell/MicroSwitch	Smart Control	NT/INtime
Honeywell IAC	Total Plant Solution	NT
Iconics	Genesis32	NT
Imagination Systems	Hyperkernel	Real-time system
Integrated Systems Inc.	Psos, psos +	Real-time system, tools
Intellution	FIX Paradym-31	NT (RTX), CE
Intrinsyc Software	Integration Expert for Windows NT	NT, CE Development Tools
Intuitive Technology	Web@aGlance server	Java
Klopper & Wiege	ProConOS	NT with real-time kernel
Labtech	Control, Control Pro	DOS, Windows 3.1, 95
Lynx	LynxOS	Rtos
MDSI	Open CNC	NT with RTX/QNX

PC Control Vendors		
Company	**Software**	**Strategy**
Mitsubishi Electric Automation	MELsoft	NT/Intime
Moore Automation	APACS Process Control System	NT
National Instruments	Bridgeview, Labview RT	NT, NT with PC board
NemaSoft	OpenControl, Paragon	NT/Hyperkernel, NT
New Monics	PERC Virtual Machine	NT/RTX/Java
Object Automation	OAenterprise 98	NT/RTX/Java
OMNX	OMNX Direct Control	NT/QNX
Omron	Sysmac	PLC on PC board
Opto 22	OptoRuntimePC, Factory Floor	NT
OSI	PI System	NT
PC Soft	WizPLC, WizDCS	NT/RTX, NT
PEP Modular Computers	Smart2 PLC	NT
PharLap	Real-time ETS kernel for NT	Real-time systems, tools
Phoenix Contact	Remote Field Controller	Rtos, CE (in beta)
QNX Software Systems	QNX	Real-time systems
Radisys	INtime	Real-time systems
Rockwell Automation	Allen-Bradley SoftLogix 5	NT
Rockwell Software	ControlPak	NT
Schneider Automation	TSX Premium	PLC on PC board
Sequentia	OpenBatch	NT
Siemens	WinAC, MP270	NT, CE
Softing	4 Control	NT
SoftPLC	SoftPLC	Embedded RTOS/Java
Steeplechase	Visual Logic Controller	NT/Intime
TA Engineering	AIMAX	NT
Think & Do	Think & Do	NT/RTX
Total Control Products	FrameworX	NT/CE/RTX
VenturCom	RTX	Real-time systems
VMIC	IOWorks NT/VxWorks	NT/Rtos
Westinghouse PC	Web Access View Enabler (WAVE)	Java
Wind River Systems	VxWorks	Real-time systems
Wonderware	FactorySuite 2000, InControl	NT, CE
Yokogawa Industrial Automation	Centum CS 1000	NT

Características da Norma IEC 61131-3

Convenção de nomes

Endereçamento simbólico → Endereçamento físico



DIGESTOR_1
DIGESTOR_1.SEQUÊNCIA
DIGESTOR_1.SEQUÊNCIA.FATOR_H
DIGESTOR_1.SEQUÊNCIA.FATOR_H.CONTROLE_TEMP
DIGESTOR_1.SEQUÊNCIA.FATOR_H.CONTROLE_TEMP.TIC101_VP

17051021

Convenção de nomes

- Tornam o banco de dados e a estratégia de controle mais claros e perceptíveis
- Segue hierarquia *top-down*
- Nome embutido no nível de recurso, o computador ou sistema externo solicita dado pelo nome ao invés de buscar endereço
- Facilita integração com outros sistemas: administrativo, gestão
- Facilita e exportação da base de dados
- Uso de Literais: -12, 0.45, 2#1010, FALSE, TRUE, Time#14ms
- Etc

Características

Tipos de dados

- Elementares (Bool, INT, Real, Time, Date, String, Byte, Word, etc…)
- Genéricos
  - ANY
  - ANY - NUM
  - ANY - BIT
  - STRING
  - ANY - DATE
  - TIME
  - DERIVADOS
- Derivados (*enumerated, subrange, array, struct)*

Características

Variáveis (Declaração, Atribuição de tipos, Inicialização)

- **Elemento simples**
  Consiste de um único elemento de um tipo de dado
  Ex:

%QX75 e %Q75	**Bit de saída 75**
%IW215	**Palavra de entrada 215**
%IW2.5.7.1 (hierárquico)	**Canal 1, módulo 7, rack 5, barramento 2**

- **Múltiplos elementos**
  - ***Array*** **- Coleção de dados do mesmo tipo**
    Ex: Vetores, matrizes
  - ***Structure*** **- Estrutura de dados (Registro)**

Exemplo de variáveis

VAR
  AT %IW6.2 : WORD;
  AT %MW6 : INT ;
END_VAR
	Declaração direta (posição de memória)

VAR_GLOBAL
  LIM_SW_S5 AT %IX27 : BOOL;
  TEMPERATURE AT %IW28: INT;
END_VAR
	Declaração de variáveis simbólicas como alocação direta

VAR INARY AT %IW6 : ARRAY [0..9] OF INT; END_VAR	Declaração de array com alocação direta

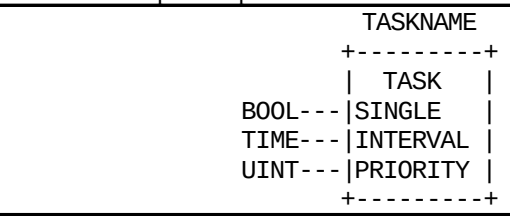
VAR
  CONDITION_RED : BOOL;
  IBOUNCE : WORD ;
  AWORD, BWORD, CWORD : INT;
  MYSTR: STRING[10] ;
END_VAR
	Alocação dinâmcia de variáveis simbólicas

TYPE PRESSURE_SENSOR :
  STRUCT
  INPUT : REAL := 2.0 ;
  STATUS : BOOL := 0 ;
  CALIBRATION : DATE := DT#1999-05-20 ;
  HIGH_LIMIT : REAL := 30.0 ;
  ALARM_COUNT : INT := 0 ;
  END_STRUCT
END_TYPE
	Declaração do tipo de dados e inicialização de variavél *structure*

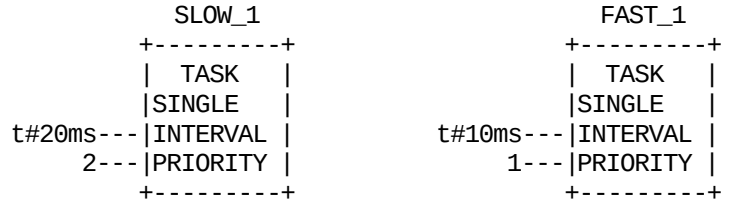
Controle de Tarefas

- **Engatilhada (*Single*)**
- **Periódica (*Interval*)**
- **Prioridades (Preemptivo ou não)**

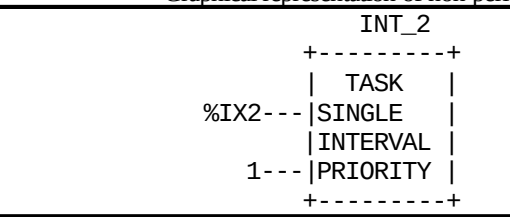
Graphical representation of TASKs within a RESOURCE



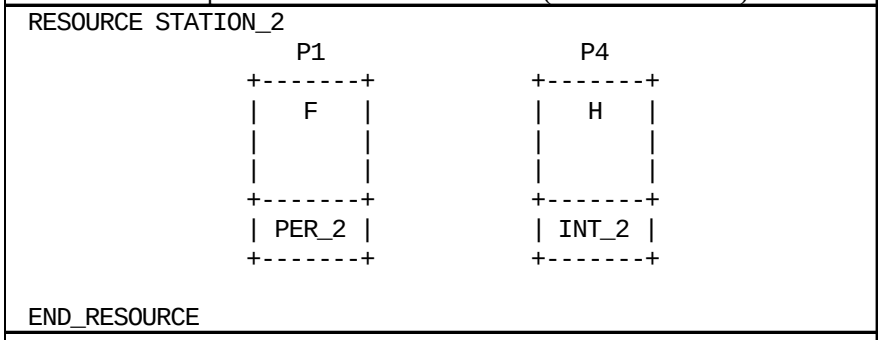
Graphical representation of periodic TASKs



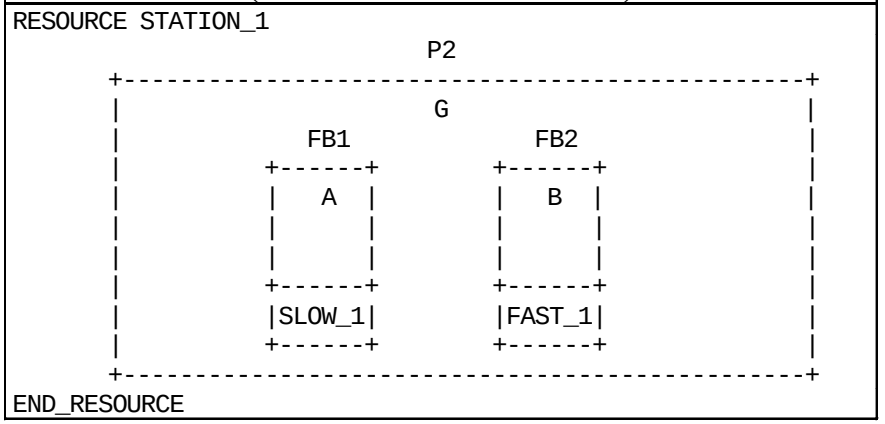
Graphical representation of non-periodic TASK



Graphical association with PROGRAMs (within RESOURCEs)



Graphical association with function blocks (within PROGRAMs inside RESOURCEs)



Unidades de Organização de Programas (POU)

- Função (*Function*)
- Bloco de Função (*Function Block*)
- Programa (*Program*)

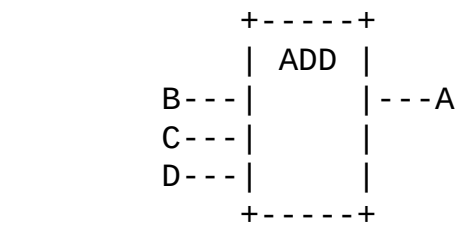
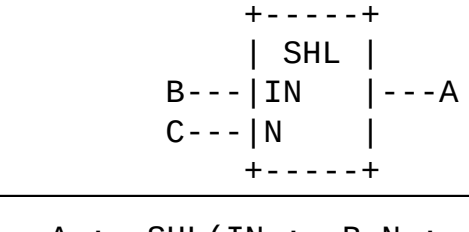
Características

- Fornecidas pelo fabricante (software, equipamentos, etc…)
- Desenvolvidas pelo usuário
- Não recursivas

Função

- Deve ser declarada
- Quando executada produz um elemento de dado simples ou múltiplo (*array ou structure*)
- Pode ser invocada nas linguagens textuais como operando
- Não possui memória de estados. Isto é, invocar uma função com os mesmos argumentos (parâmetros de entrada), sempre produzirá o mesmo valor (saída)
- *Type overloading*: a função pode trabalhar com todos os tipos de dados de um tipo genérico
- Uma função pode ser utilizada na declaração de outra POU

Exemplo de função

Example	Explanation
	Graphical use of ADD function (No formal parameter names)
`A := ADD(B,C,D);`	Textual use of ADD function
	Graphical use of SHL function (Formal parameter names)
`A := SHL(IN := B,N := C);`	Textual use of SHL function

Funções Padrões

Classe	Funções
Conversão de tipo	*_TO_**, TRUNC, BCD_TO_**, *_TO_BCD
Numéricas	ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN
Aritméticas	ADD (+), MUL (*), SUB (-), DIV (/), MOD, EXPT (**), MOVE (:=)
Deslocamento de Bit	SHL, SHR, ROR, ROL
Booleanas	AND (&), OR (>=1), XOR, NOT
Seleção	SEL, MAX, MIN, LIMIT, MUX
Comparação	GT (>), GE (>=), EQ (=), LE (<=), LT (<), NE (<>)
String	LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND
Tempo	ADD (+), SUB (-), MUL (*), DIV (/), CONCAT, DATE_AND_TIME_TO_TIME_OF_DAY, DATE_AND_TIME_TO_DATE
Enumerações	SEL, MUX, EQ, NE

Funções Derivadas

Podem ser criadas usando-se

- **Funções padrões ou derivadas**
- **Tipos de dados padrões ou derivados**
- **Qualquer uma das linguagens IL, ST, LD, ou FBD**
- **Linguagens adicionais**

Funções Derivadas - Exemplo

A textual form of the declaration of this function is:

```
FUNCTION WEIGH : WORD     (* BCD encoded *)
  VAR_INPUT  (* "EN" input is used to indicate "scale ready" *)
    weigh_command : BOOL;
    gross_weight : WORD ; (* BCD encoded *)
    tare_weight : INT ;
  END_VAR
(* Function Body *)
END_FUNCTION                        (* Implicit "ENO" *)
```

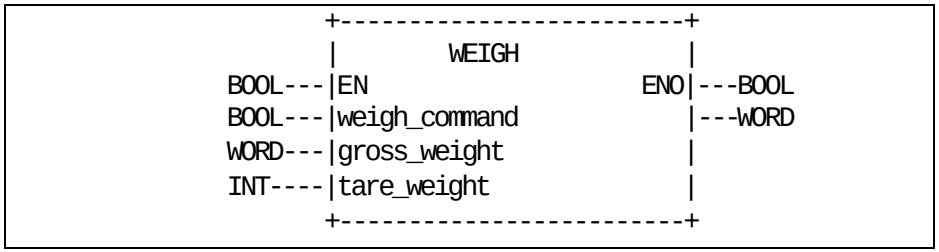
The body of function WEIGH in the IL language is:

```
              LD          weigh_command
              JMPC        WEIGH_NOW
              ST          ENO          (* No weighing, 0 to "ENO" *)
              RET
WEIGH_NOW:    LD          gross_weight
              BCD_TO_INT
              SUB         tare_weight
              INT_TO_BCD               (* Return evaluated weight *)
              ST          WEIGH
```

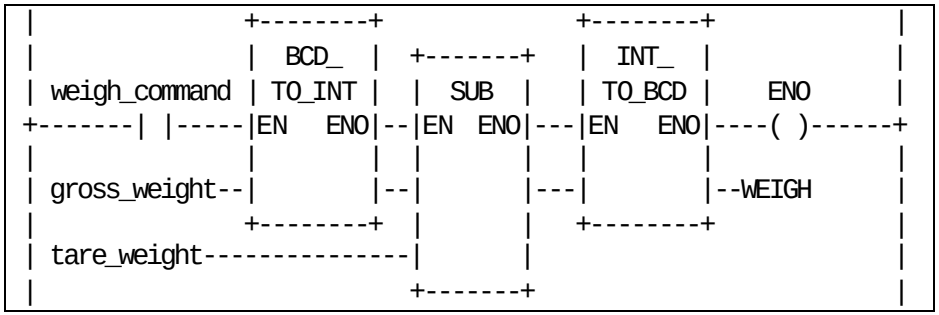
The body of function WEIGH in the ST language is:

```
IF weigh_command THEN
  WEIGH := INT_TO_BCD (BCD_TO_INT(gross_weight) - tare_weight);
END_IF ;
```

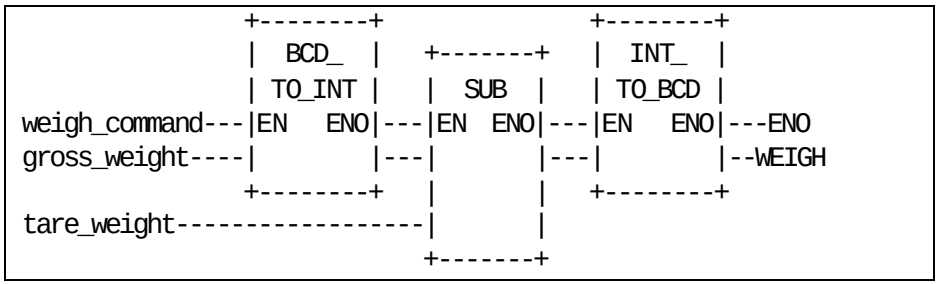
An equivalent graphical declaration of function WEIGH is:



The function body in the LD language is:

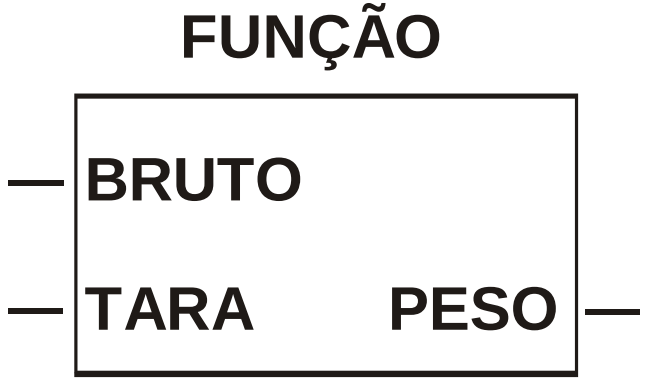


The function body in the FBD language is:



Funções Derivadas – Exemplo IsaGraf

Representação gráfica



Lista de Instruções

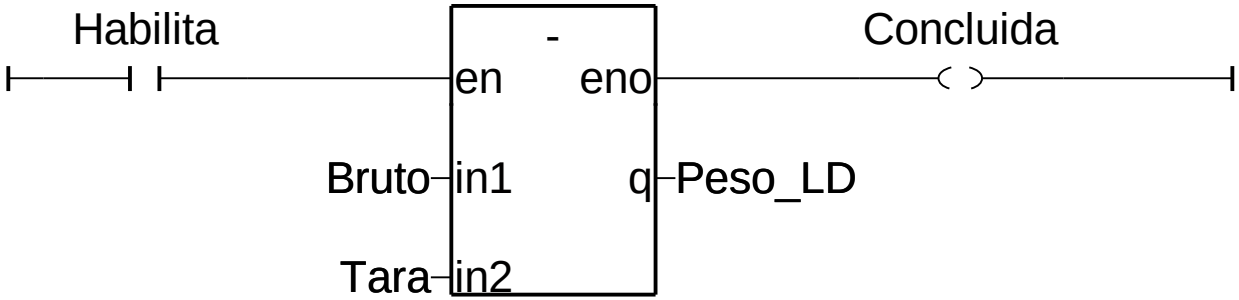
```
LD Bruto
SUB Tara
ST Peso_IL
```

Texto Estruturado

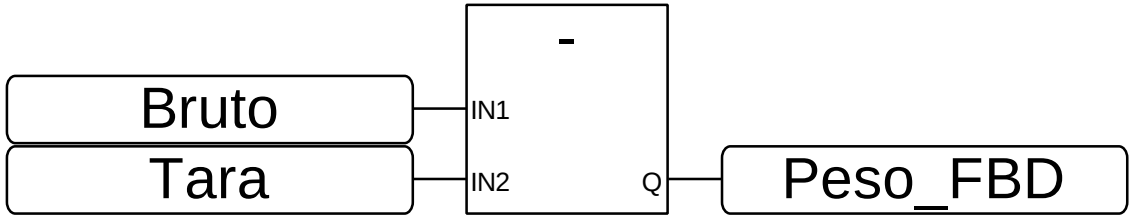
```
Peso_ST := Bruto - Tara
```

Funções Derivadas – Exemplo IsaGraf

Ladder



Function Block



Bloco de Função

- Deve ser declarado
- Quando executado produz um ou mais valores
- Instanciação: várias instâncias (identificador + estrutura de dados) podem ser criadas
- Possui memória de dados: todas as variáveis internas e de saída são mantidas entre as execuções de um bloco de função
- Pode ser invocado pelas linguagens textuais
- Um bloco de função pode ser usado na declaração de outro bloco de função ou programa (instanciação)

Exemplo de bloco de função

```
FUNCTION_BLOCK DEBOUNCE
(*** External Interface ***)
VAR_INPUT
  IN : BOOL ;                  (* Default = 0 *)
  DB_TIME : TIME := t#10ms ;   (* Default = t#10ms *)
END_VAR
VAR_OUTPUT OUT : BOOL ;        (* Default = 0 *)
    ET_OFF : TIME ;            (* Default = t#0s *)
END_VAR
VAR DB_ON : TON ;              (** Internal Variables **)
    DB_OFF : TON ;             (**  and FB Instances  **)
    DB_FF : SR ;
END_VAR

(** Function Block Body **)
DB_ON(IN := IN, PT := DB_TIME) ;
DB_OFF(IN := NOT IN, PT:=DB_TIME) ;
DB_FF(S1 :=DB_ON.Q, R := DB_OFF.Q) ;
OUT := DB_FF.Q ;
ET_OFF := DB_OFF.ET ;

END_FUNCTION_BLOCK
```

```
FUNCTION_BLOCK
(** External Interface **)
                +---------------+
                |   DEBOUNCE    |
        BOOL---|IN          OUT|---BOOL
        TIME---|DB_TIME  ET_OFF|---TIME
                +---------------+
(** Function Block Body **)

                DB_ON       DB_FF
               +-----+     +----+
               | TON |     | SR |
 IN----+------|IN  Q|-----|S1 Q|---OUT
       |  +---|PT ET|  +--|R   |
       |  |   +-----+  |  +----+
       |  |            |
       |  |    DB_OFF  |
       |  |   +-----+  |
       |  |   | TON |  |
      +--|--O|IN  Q|--+
 DB_TIME--+---|PT ET|--------------ET_OFF
              +-----+
 END_FUNCTION_BLOCK
```

Blocos de Função Padrões

Biestáveis

```
       +-----+
       |  SR |
 BOOL---|S1 Q1|---BOOL
 BOOL---|R    |
       +-----+
```

```
       +-----+
       |  RS |
 BOOL---|S  Q1|---BOOL
 BOOL---|R1   |
       +-----+
```

Detecção de Borda

```
       +--------+
       | R_TRIG |
BOOL---|CLK    Q|---BOOL
       +--------+
```

```
       +--------+
       | F_TRIG |
BOOL---|CLK    Q|---BOOL
       +--------+
```

Contagem

```
       +-----+
       | CTU |
 BOOL--->CU  Q|---BOOL
 BOOL---|R    |
  INT---|PV CV|---INT
       +-----+
```

```
       +-----+
       | CTD |
 BOOL--->CD  Q|---BOOL
 BOOL---|LD   |
  INT---|PV CV|---INT
       +-----+
```

```
       +------+
       | CTUD |
 BOOL--->CU  QU|---BOOL
 BOOL--->CD  QD|---BOOL
 BOOL---|R     |
 BOOL---|LD    |
  INT---|PV  CV|---INT
       +------+
```

Temporização

```
       +-------+
       |  TON  |
 BOOL---|IN    Q|---BOOL
 TIME---|PT   ET|---TIME
       +-------+
```

```
       +-------+
       |  TOF  |
 BOOL---|IN    Q|---BOOL
 TIME---|PT   ET|---TIME
       +-------+
```

```
       +-------+
       |  TP   |
 BOOL---|IN    Q|---BOOL
 TIME---|PT   ET|---TIME
       +-------+
```

```
       +-------+
       |  RTC  |
 BOOL---|IN    Q|---BOOL
 DT-----|PDT CDT|-----DT
       +-------+
```

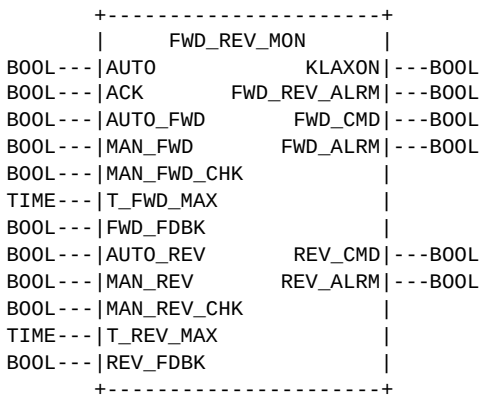
Comunicação (IEC 61131-5)	
Conexão	CONNECT
Verificação de Dispositivos	STATUS, USTATUS
Aquisição de Dados	READ, USEND, URCV
Controle	WRITE, SEND, RCV
Alarmes	NOTIFY, ALARM

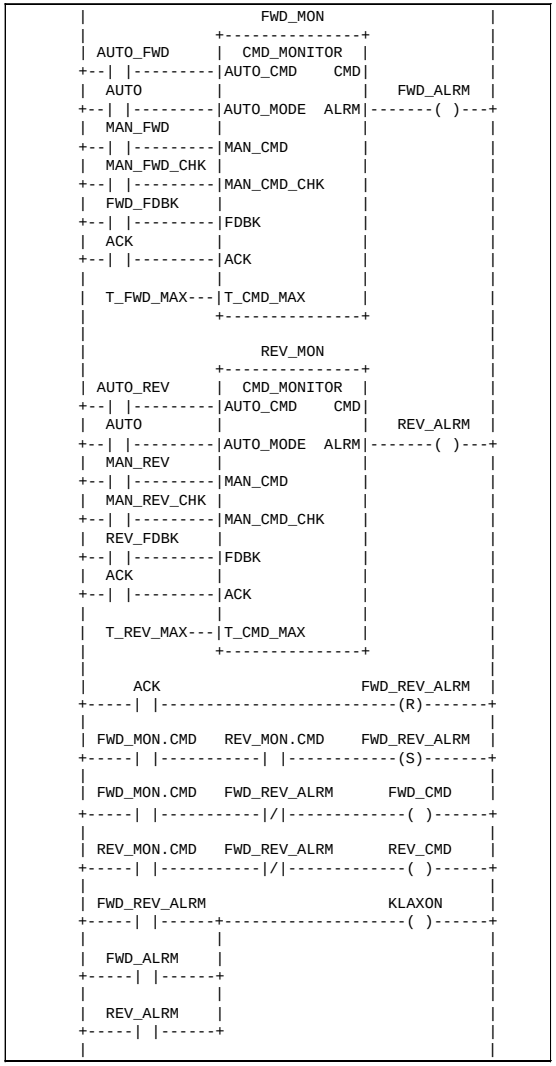
Blocos de Função Derivados

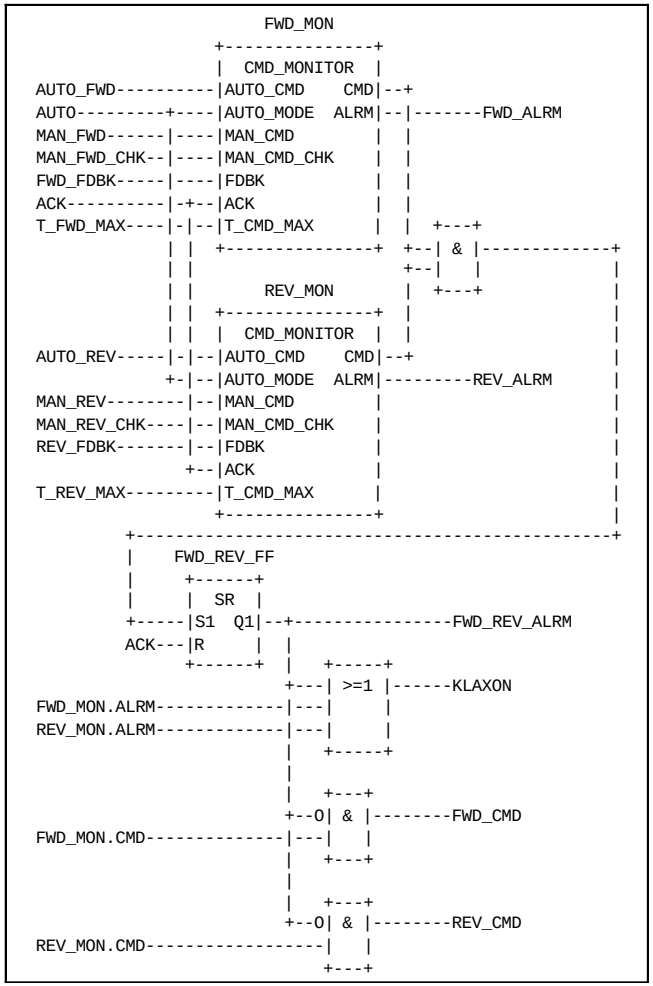
Podem ser criados usando-se

- **Blocos de função padrões ou derivados**
- **Tipos de dados padrões ou derivados**
- **Qualquer uma das linguagens: IL, ST, LD, FBD ou SFC**
- **Linguagens adicionais**

Blocos de Função Derivados
Exemplo





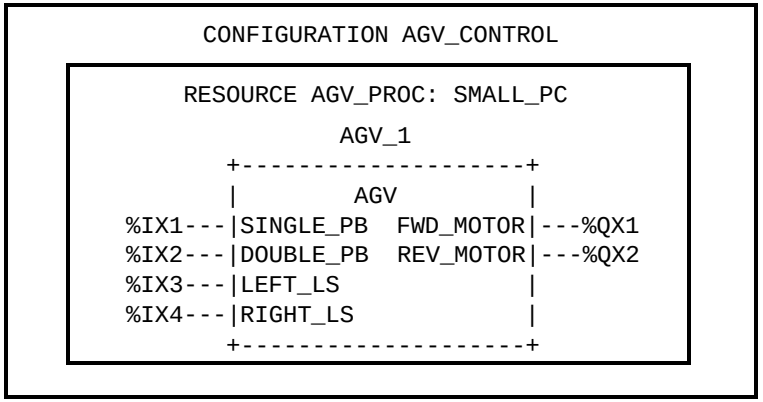


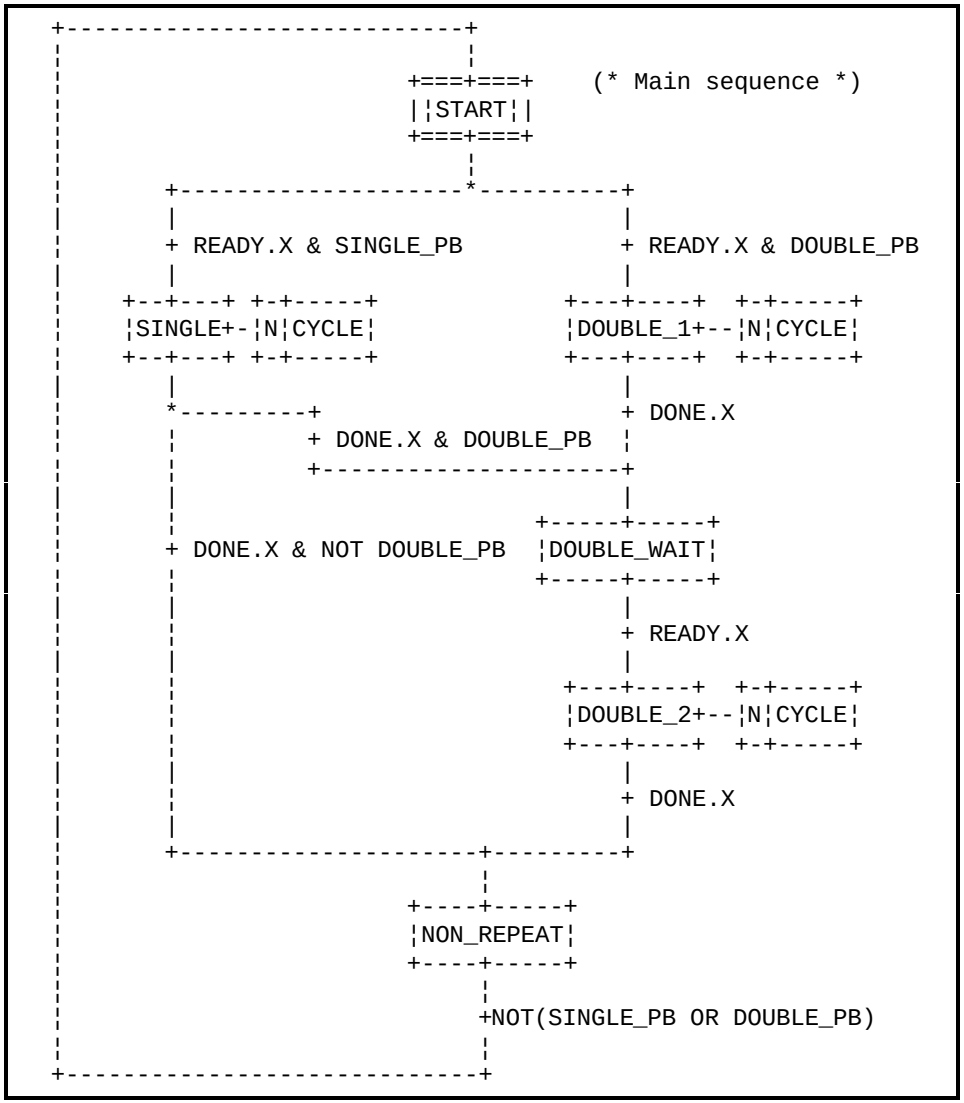
Programa

É um agrupamento lógico dos elementos necessários à todas as linguagens de programação, para o processamento de sinais desejado

- **Deve ser declarado**
- **Permite a instanciação somente dentro de um recurso**

Exemplo de programa

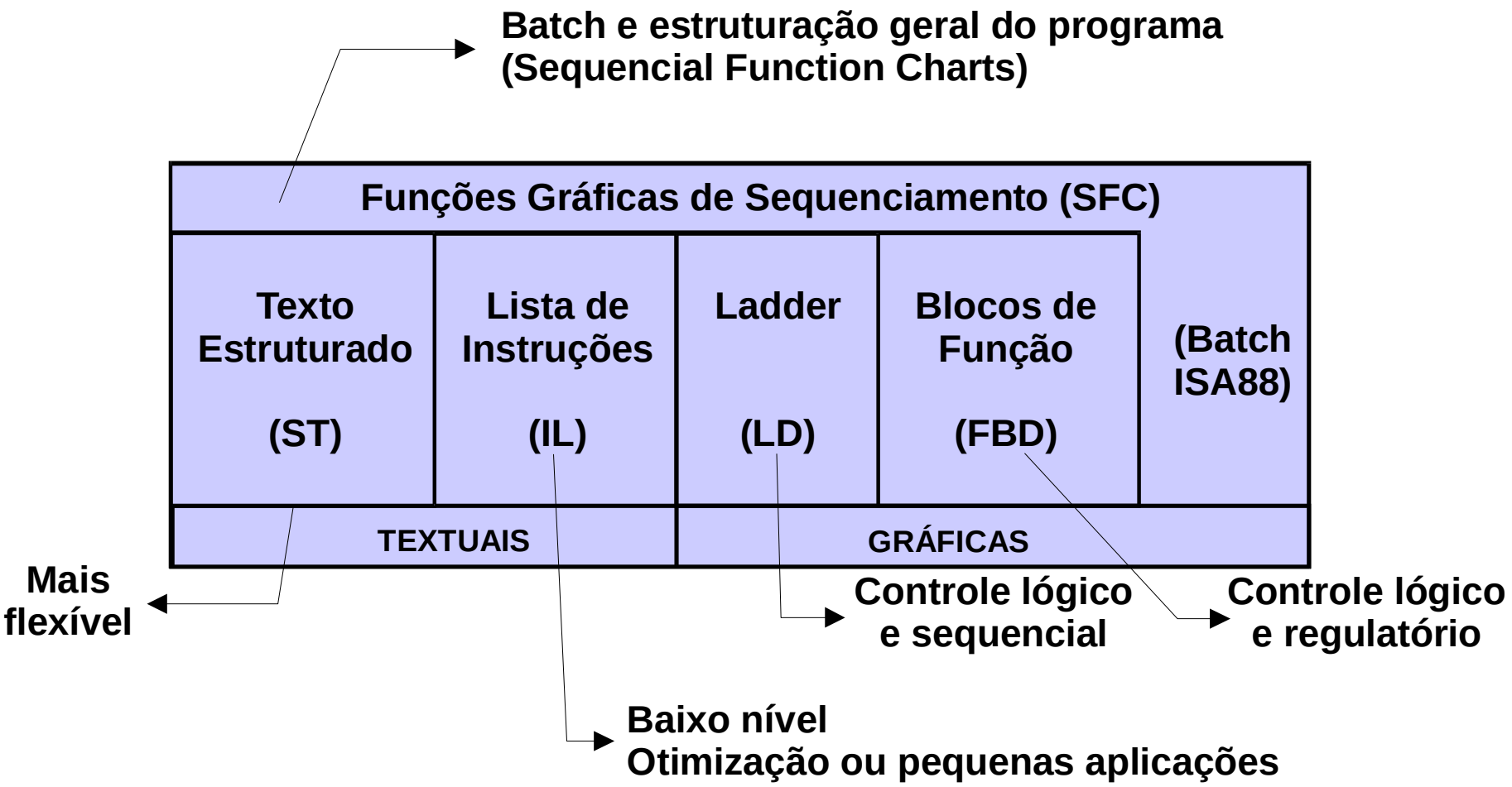




Linguagens de Programação

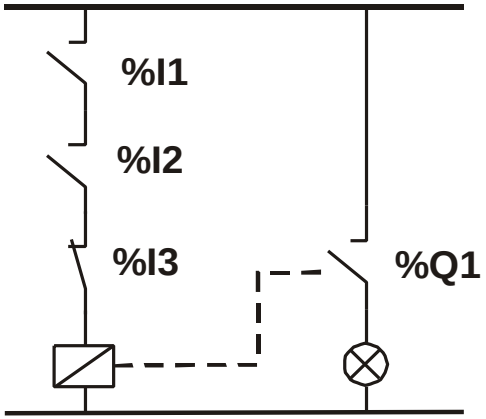
- Textuais
  - IL - Lista de Instruções
  - ST - Texto Estruturado
- Gráficas
  - LD - *Ladder*
  - FBD - Diagrama de Blocos de Função
- Organização de Programas
  - SFC - Funções Gráficas de Seqüenciamento
- Outras (dependente do produto)
  - *Flow Chart*
  - C
  - Etc.

Linguagens

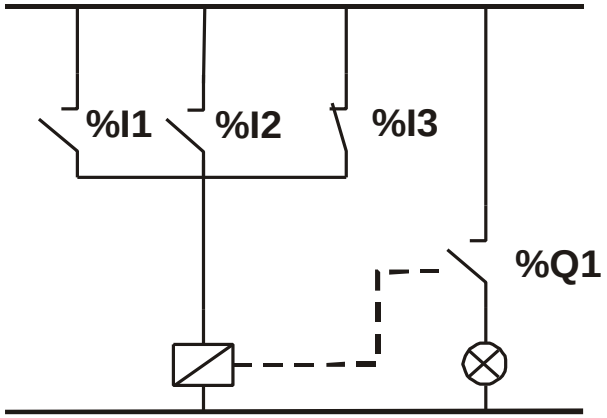

Batch e estruturação geral do programa
(Sequencial Function Charts)
Funções Gráficas de Sequenciamento (SFC)
Texto Estruturado
(ST)
Lista de Instruções
(IL)
Ladder
(LD)
Blocos de Função
(FBD)
(Batch ISA88)
TEXTUAIS
GRÁFICAS
Mais flexível
Controle lógico e sequencial
Controle lógico e regulatório
Baixo nível
Otimização ou pequenas aplicações

Programação Convencional de CLPs

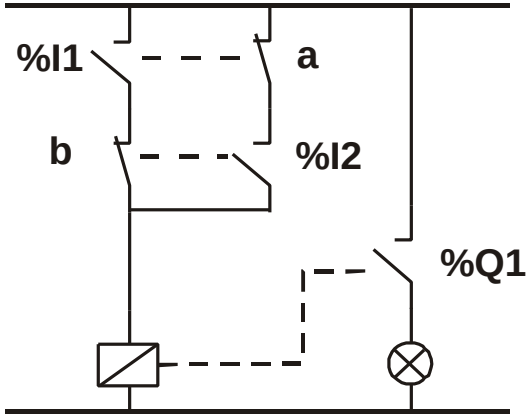
Lógica E



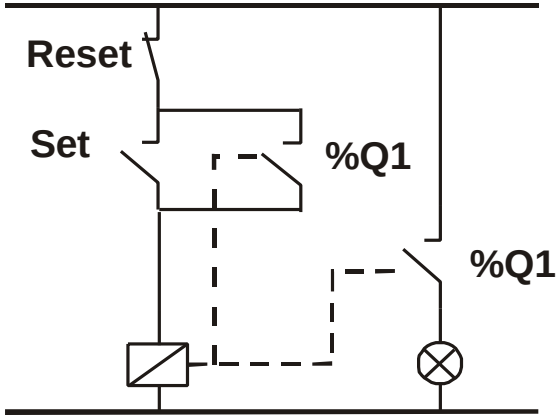
Lógica OU



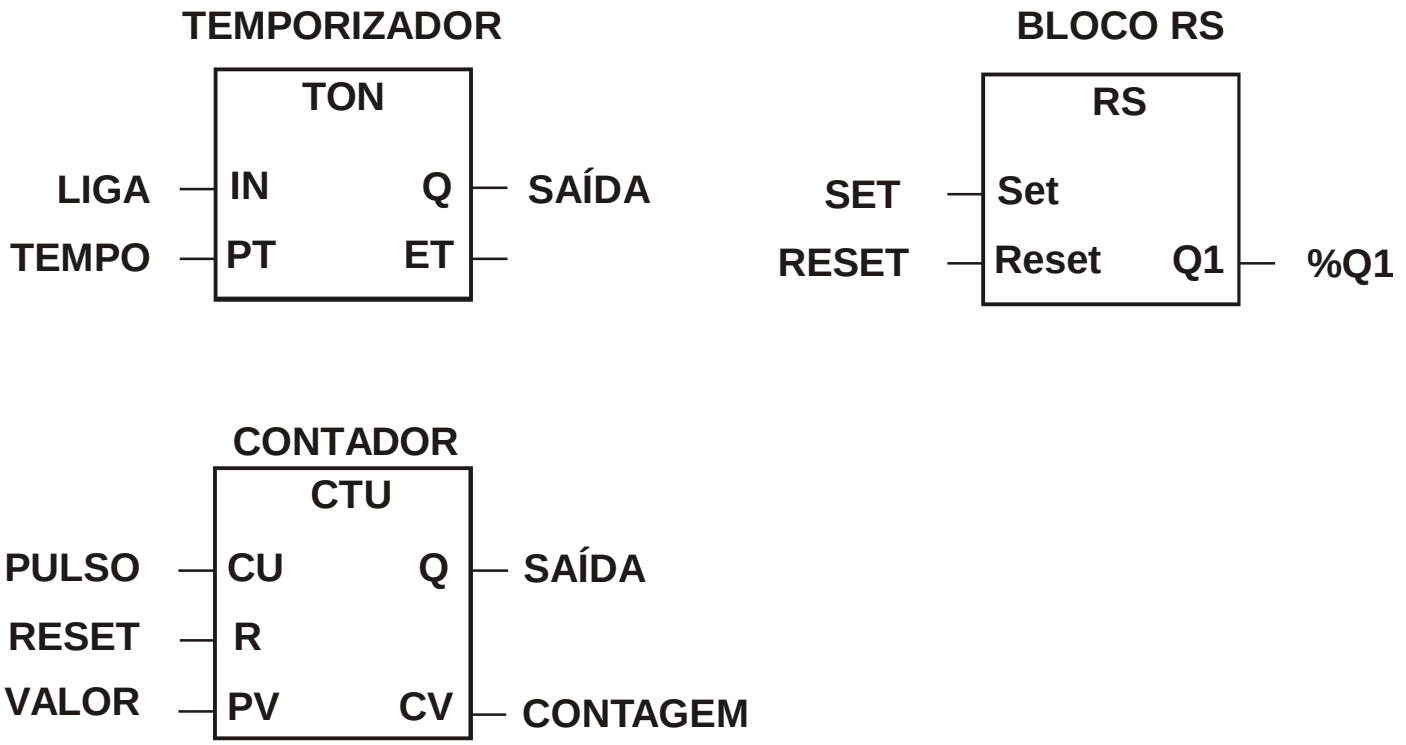
Lógica OU Exclusivo



Memória RS



Programação Convencional de CLPs



IL - Lista de Instruções

Características

- Linguagem de Baixo Nível
- Semelhante ao *Assembler*
- Recomendada para pequenas aplicações ou otimização de código
- Linguagem básica para exportação de programas (Portabilidade)

IL - Lista de Instruções

Estrutura

Label	Operator	Operand	Comment
START:	LD	%IX1	(* PUSH BUTTON *)
	ANDN	%MX5	(* NOT INHIBITED *)
	ST	%QX2	(* FAN ON *)

IL - *Instruction List*

Operadores

Operator	Modifiers	Operand	Semantics
LD	N	Note 2	Set current result equal to operand
ST	N	Note 2	Store current result to operand location
S	Note 3	BOOL	Set Boolean operand to 1
R	Note 3	BOOL	Reset Boolean operand to 0
AND	N, (	BOOL	Boolean AND
&	N, (	BOOL	Boolean AND
OR	N, (	BOOL	Boolean OR
XOR	N, (	BOOL	Boolean Exclusive OR
ADD	(	Note 2	Addition
SUB	(	Note 2	Subtraction
MUL	(	Note 2	Multiplication
DIV	(	Note 2	Division
GT	(	Note 2	Comparison: >
GE	(	Note 2	Comparison: >=
EQ	(	Note 2	Comparison: =
NE	(	Note 2	Comparison: <>
LE	(	Note 2	Comparison: <=
LT	(	Note 2	Comparison: <
JMP	C, N	LABEL	Jump to label
CAL	C, N	NAME	Call function block (note 4)
RET	C, N		Return from called function or function block
)			Evaluate deferred operation

Operadores para Blocos de Função Padrões

Operators	FB Type
S1,R	SR
S,R1	RS
CLK	R_TRIG
CLK	F_TRIG
CU,R,PV	CTU
CD,LD,PV	CTD
CU,CD,R,LD,PV	CTUD
IN,PT	TP
IN,PT	TON
IN,PT	TOF

IL - *Instruction List* - Exemplos IsaGraf

AND	OR	XOR
LD %IX0.1 AND %IX0.2 ANDN %IX0.3 ST %QX1.1	LD %IX0.1 OR %IX0.2 ORN %IX0.3 ST %QX1.1	LD %IX0.1 XOR %IX0.2 ST %QX1.1
Memória RS	**Temporizador TON**	**Contador UP**
LDN Reset AND(Set OR %QX1.1) ST %QX1.1	LD Liga ST Timer.in LD Tempo ST Timer.pt CAL Timer LD Timer.q ST Saida	LD Pulso ST Cont1.cu LD Reset ST Cont1.reset LD Valor ST Cont1.pv CAL Cont1 LD Cont1.Q ST Saida

ST - Texto Estruturado

Características

- Linguagem de alto nível
- Semelhante ao Pascal (ISO 7185)
- Ideal para
  - Tomada de decisões
  - Declarações (Variáveis, POUs, Configurações, etc.)
  - Cálculos
  - Implementação de algoritmos
  - Definição de ações (SFC)
  - Utilização de literais
  - Criação de blocos
  - Etc.

ST - *Structured Text*

Operadores

Operation	Symbol	Precedence
Parenthesization	(expression)	HIGHEST
Function evaluation **Examples**:	identifier (argument list) LN(A), MAX(X,Y), etc.	
Exponentiation (Note 2)	**	
Negation Complement	- NOT	
Multiply Divide Modulo	* / MOD	
Add Subtract	+ -	
Comparison	< , > , <= , >=	
Equality Inequality	= <>	
Boolean AND Boolean AND	& AND	
Boolean Exclusive OR	XOR	
Boolean OR	OR	LOWEST

Declarações

Statement type/Reference	Examples
Assignment	`A := B;  CV := CV+1; C := SIN(X);`
Function block Invocation and FB output usage	`CMD_TMR(IN:=%IX5, PT:=T#300ms) ;` `A := CMD_TMR.Q ;`
RETURN	`RETURN ;`

IF	`D := B*B - 4*A*C ;`
	`IF D < 0.0 THEN NROOTS := 0 ;`
	`ELSIF D = 0.0 THEN`
	`  NROOTS := 1 ;`
	`  X1 := - B/(2.0*A) ;`
	`ELSE`
	`  NROOTS := 2 ;`
	`  X1 := (- B + SQRT(D))/(2.0*A) ;`
	`  X2 := (- B - SQRT(D))/(2.0*A) ;`
	`END_IF ;`

CASE	`TW := BCD_TO_INT(THUMBWHEEL);`
	`TW_ERROR := 0;`
	`CASE TW OF`
	`  1,5:  DISPLAY := OVEN_TEMP;`
	`  2:  DISPLAY := MOTOR_SPEED;`
	`  3:  DISPLAY := GROSS - TARE;`
	`  4,6..10: DISPLAY := STATUS(TW - 4);`
	`ELSE  DISPLAY := 0 ;`
	`  TW_ERROR := 1;`
	`END_CASE;`
	`QW100 := INT_TO_BCD(DISPLAY);`

FOR	`J := 101 ;`
	`FOR I := 1 TO 100 BY 2 DO`
	`  IF WORDS[I] = 'KEY' THEN`
	`  J := I ;`
	`  EXIT ;`
	`  END_IF ;`
	`END_FOR ;`

WHILE	`J := 1;`
	`WHILE J <= 100 & WORDS[J] <> 'KEY' DO`
	`  J := J+2 ;`
	`END_WHILE ;`

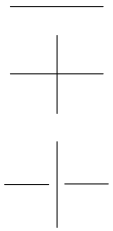
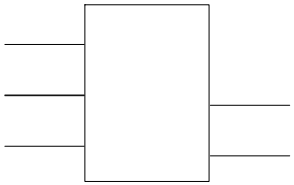
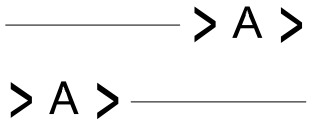
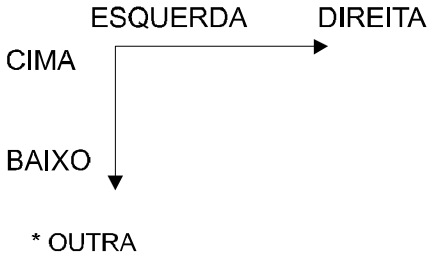
REPEAT	`J := -1 ;`
	`REPEAT`
	`  J := J+2 ;`
	`UNTIL J = 101 OR WORDS[J] = 'KEY'`
	`END_REPEAT ;`

EXIT	`EXIT ;`
Empty Statement	`;`

ST - *Structured Text* - Exemplos IsaGraf

AND	%QX1.1:= %IX0.1 AND %IX0.2 AND Not %IX0.3;
OR	%QX1.1:=%IX0.1 OR %IX0.2 OR NOT %IX0.3;
XOR	%QX1.1:=%IX0.1 XOR %IX0.2;
Memória RS	%QX1.1:= NOT Reset AND (Set OR %QX1.1);
Temporizador TON	Timer (Liga, Tempo); Saida2:= Timer.q;
ContadorUP	Cont2 (Pulso, Reset, Valor); Saida2:= Cont2.q; Resultado:= Cont2.cv;

Linguagens Gráficas - Elementos comuns

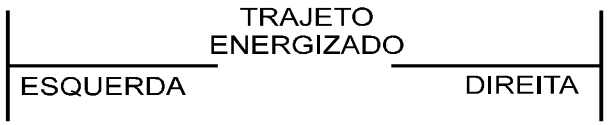
<table>
<tr><th>LINHAS E CONEXÕES</th><th>BLOCO</th><th>CONECTOR</th></tr>
<tr><td>  </td><td>  </td><td> 
</td></tr>
<tr><td>DIREÇÃO DO
FLUXO DE
CONTROLE</td><td colspan="2">TRAJETO DE ENERGIZAÇÃO (LD)
TRAJETO DO SINAL (FBD)
TRAJETO DA ATIVIDADE (SFC)
 
</td></tr>
<tr><td>DETERMINAÇÃO
DA REDE</td><td colspan="2">LEITURA DAS ENTRADAS
FIXAÇÃO DAS SAÍDAS
EXECUÇÃO DAS FUNÇÕES E BLOCOS DE FUNÇÃO
ATUALIZAÇÃO DAS SAÍDAS DA REDE
RETORNO</td></tr>
</table>

LD - *Ladder Diagram*

Características

- Baseada no diagrama elétrico de contatos
- Adequada para controle discreto, combinacional e seqüencial (intertravamento)
- Utiliza blocos de função para controle regulatório e funções especiais

LD - *Ladder Diagram*

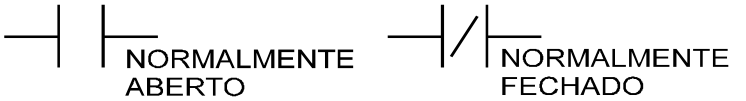


LINK HORIZONTAL

LINK VERTICAL

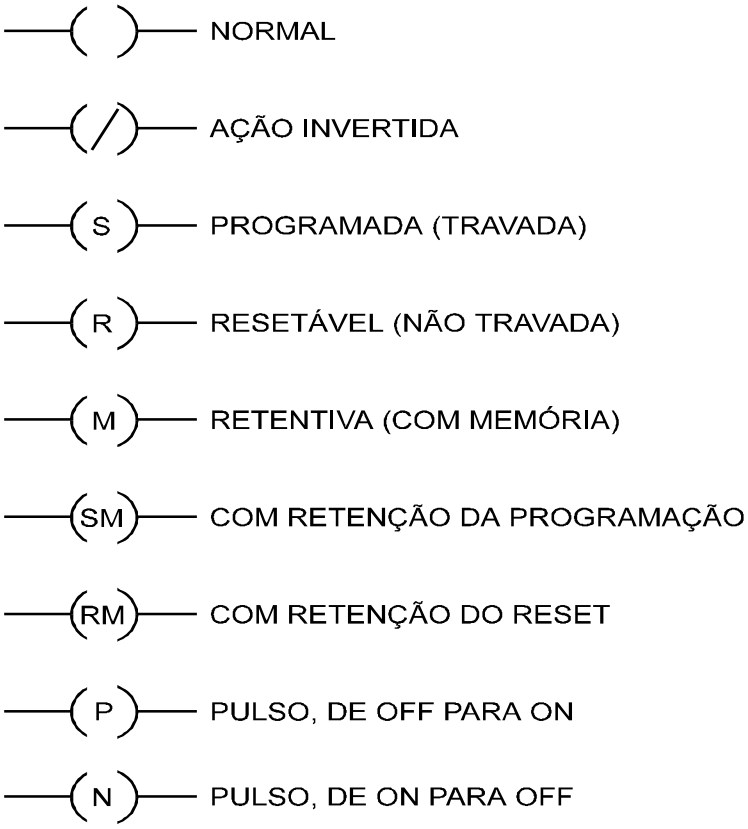
CONTATOS DE COMANDO

SIMBOLOGIA

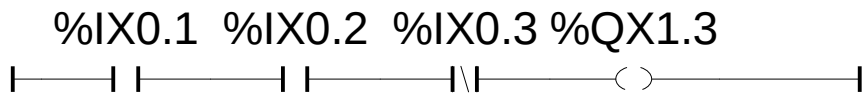
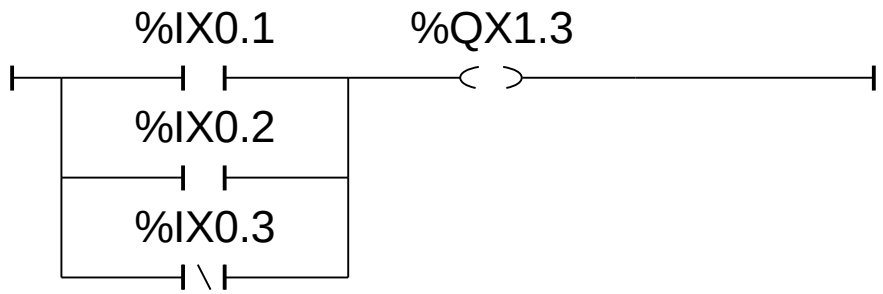
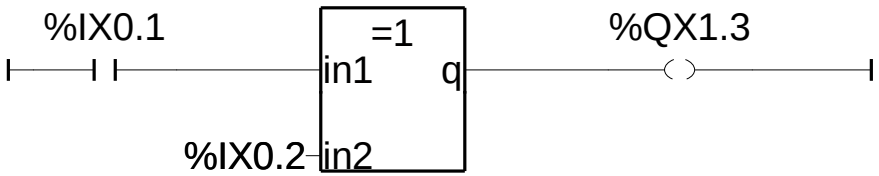


BOBINAS

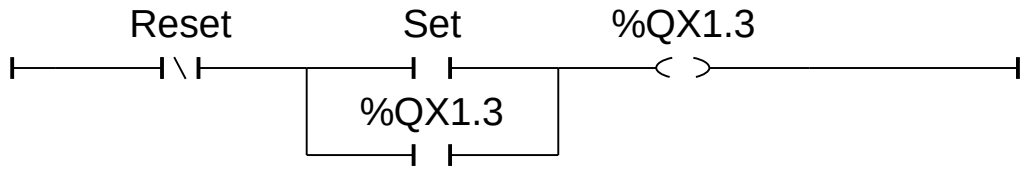
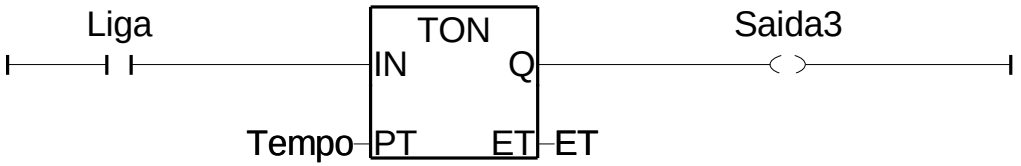
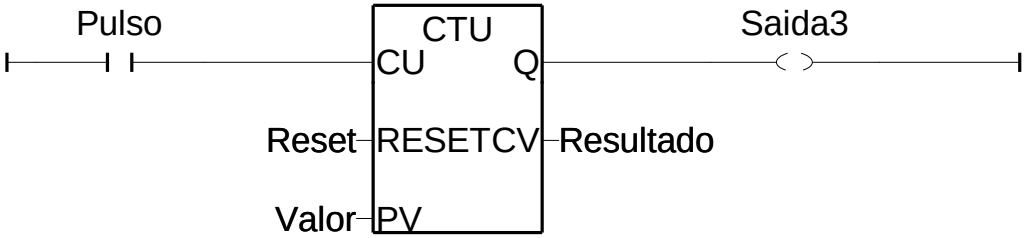
SIMBOLOGIA



LD - *Ladder Diagram* – Exemplos IsaGraf

<table>
<tr><td>AND</td><td> 
</td></tr>
<tr><td>OR</td><td> 
</td></tr>
<tr><td>XOR</td><td> 
</td></tr>
</table>

LD - *Ladder Diagram* – Exemplos IsaGraf

Memória RS	
Tempor. TON	
Contador UP	

FBD - Diagrama de Blocos Funcionais

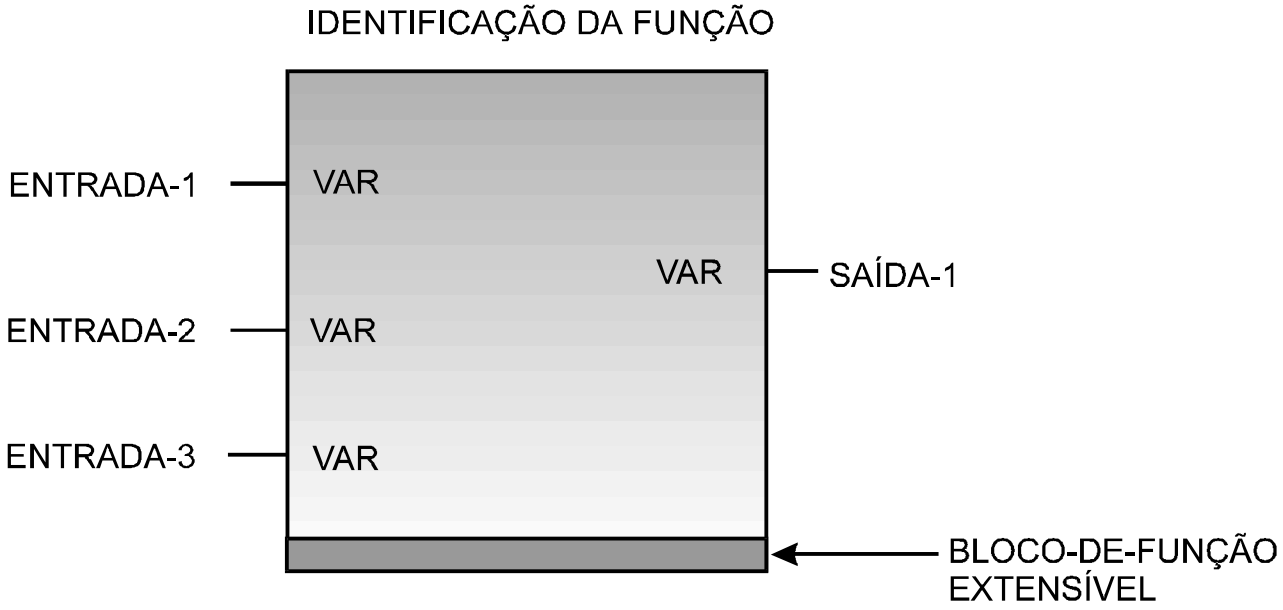
Características

- Baseada nos diagramas de circuitos (Diagrama Lógico)
- Adequada para controle discreto, seqüencial, regulatório, etc.
- Representação de fácil interpretação
- Blocos expansíveis em função do nº de parâmetros de entrada
- São disparados por parâmetros externos, enquanto os algoritmos internos permanecem escondidos (= OOP)
- Blocos encapsulam o algoritmo, destacando o fluxo de informações e o processamento de sinais

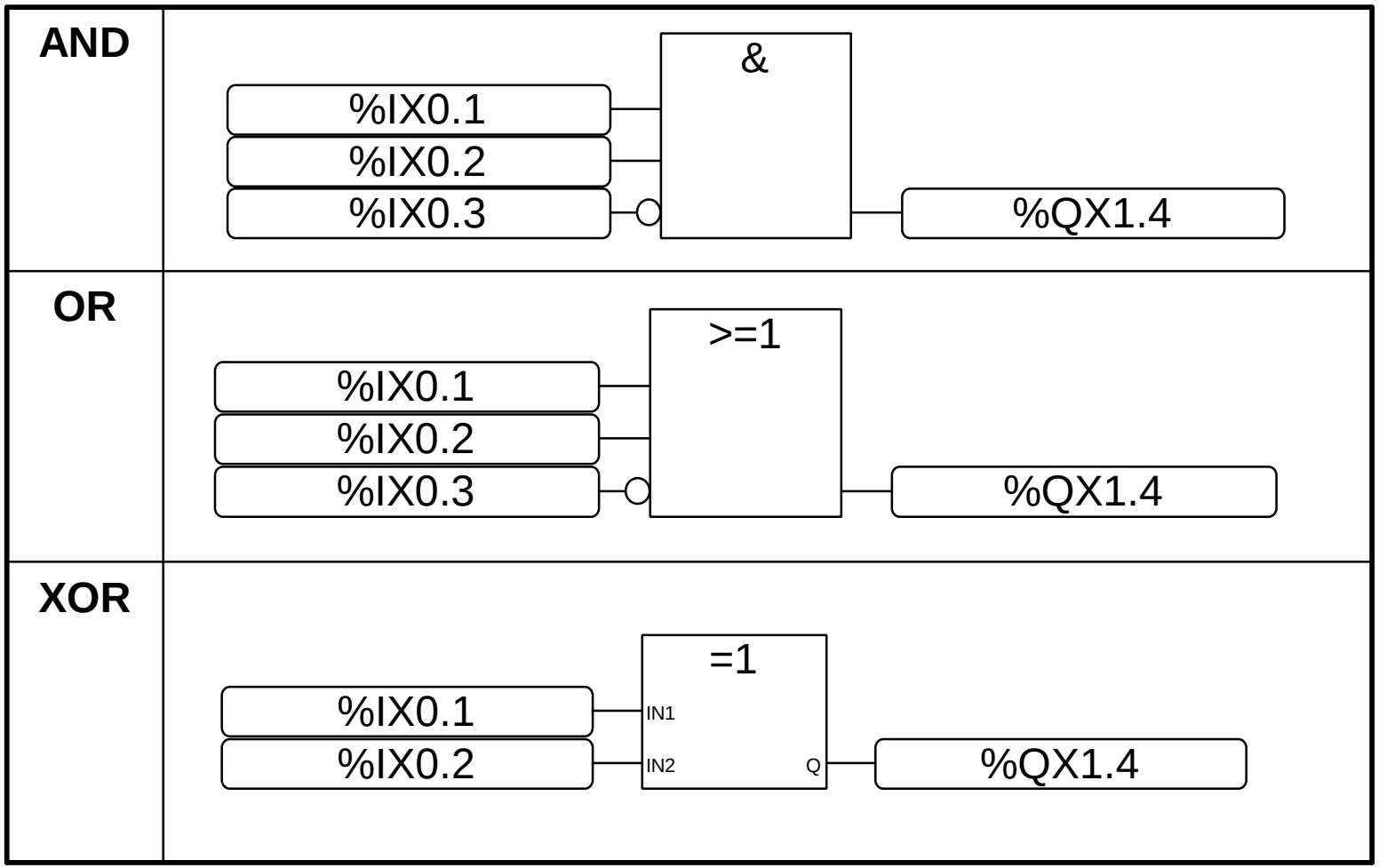
Bloco = Função ou Bloco de Função

FBD - *Function Block Diagram*

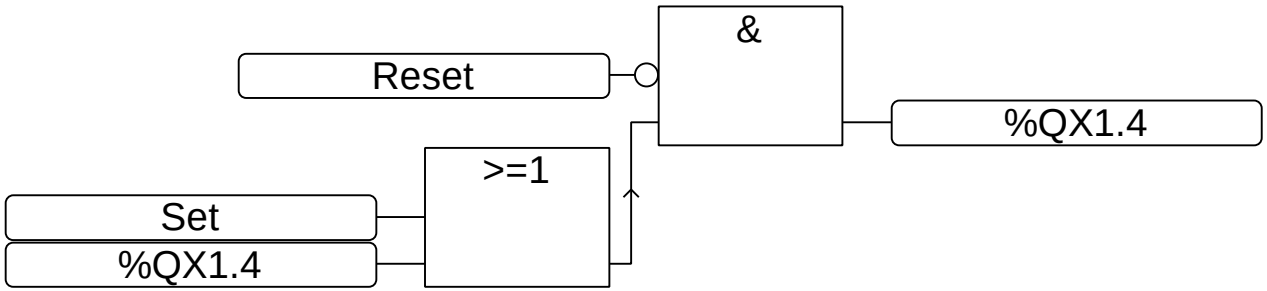
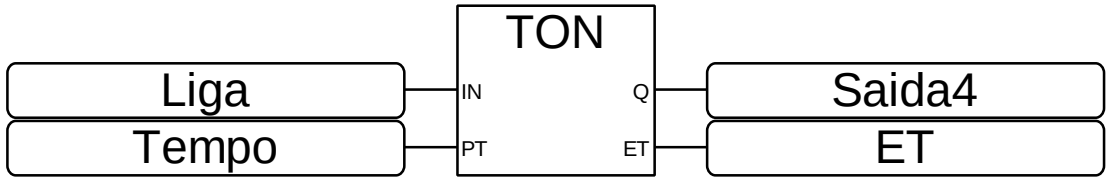
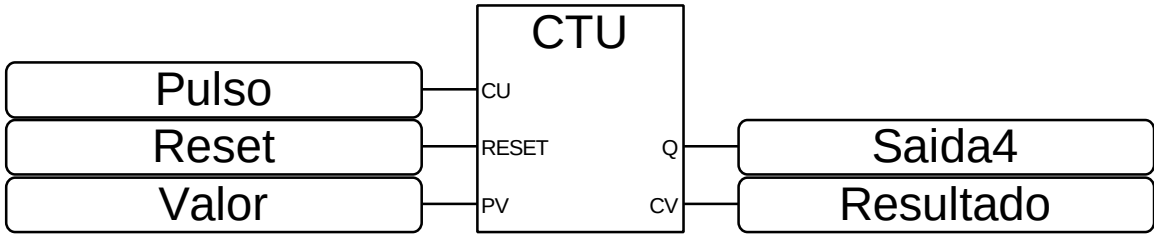
ENTRADAS	FUNÇÃO	SAÍDAS
UMA OU MAIS NÚMEROS INTEIROS REAL LÓGICO TEMPO	EQUIVALENTE AO TEXTO ESTRUTURADO	UMA OU MAIS TIPO-DEPENDENTE DA FUNÇÃO



FBD - *Function Block Diagram* – Exemplos IsaGraf



FBD - *Function Block Diagram* – Exemplos IsaGraf

Memória RS	
Tempor. TON	
Contador UP	

SFC - Funções Gráficas de Seqüenciamento

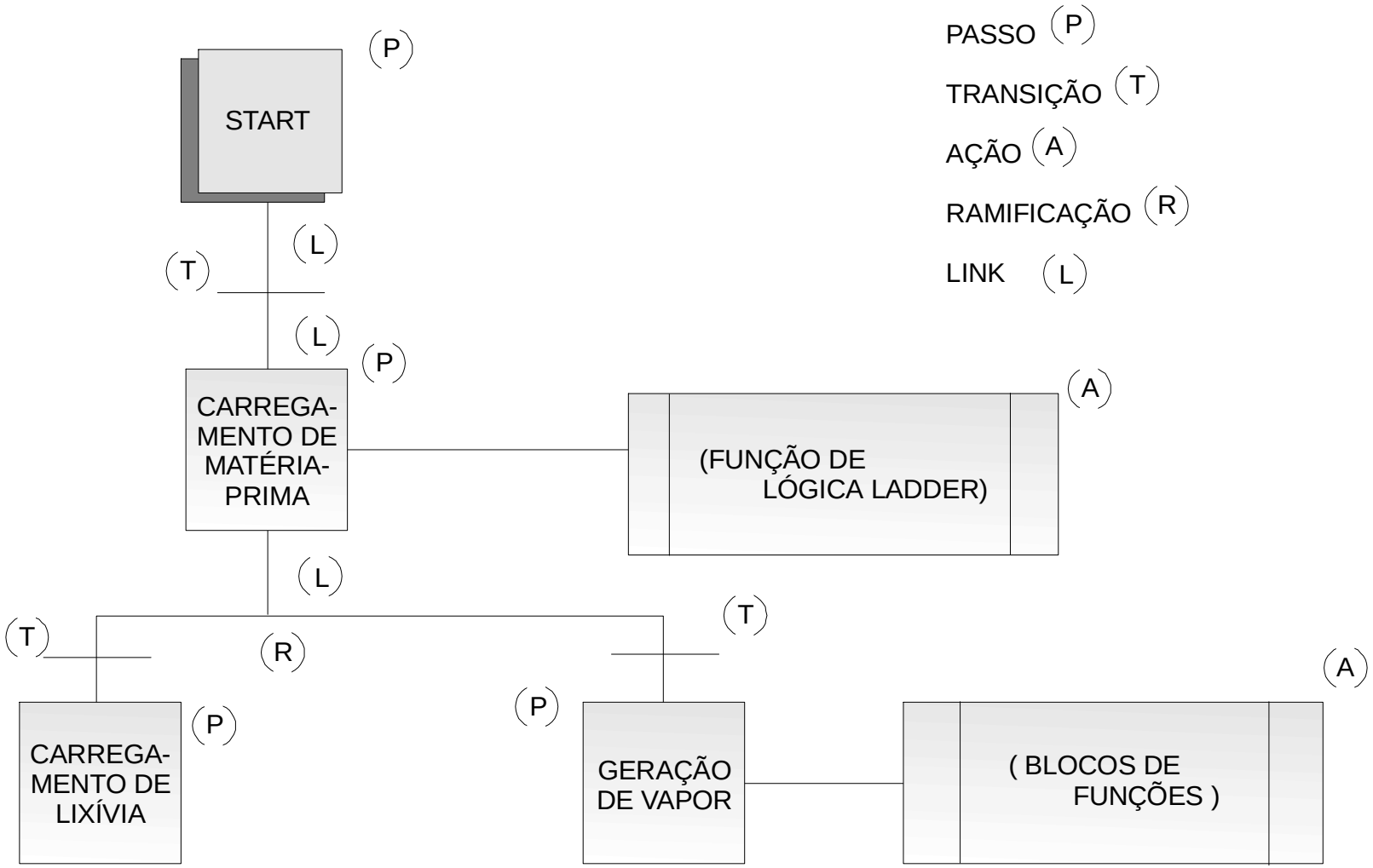
Características

- Baseada no *Grafcet* e Redes de Petri ⇨ Formulação Matemática
- Padrão para programação de processos *Batch* ⇨ ISA SP88
- Adequada para
  - Estruturação de Programas e Blocos de Função
  - Controle seqüencial ⇨ Receitas e Seqüenciamento Discreto
  - Controle de estados ⇨ Máquina de Estados Finitos e Algoritmos
  - Tomadas de decisão ⇨ Árvore de decisões
- Representação de fácil interpretação
- Rastreabilidade de eventos

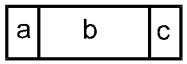
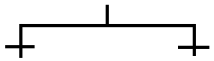
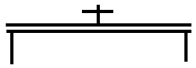
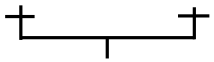
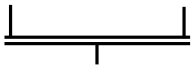
SFC - Funções Gráficas de Seqüenciamento

- Permite reduzir o esforço computacional
- Facilidade de diagnóstico
- Elementos de programação
  - Passo: estado do programa onde as ações são executadas
  - Transição: condição pela qual o programa muda de estado, passando de um ou mais passos antecessores para um ou mais passos sucessores
  - Ação: atividade de controle executada num determinado passo. Pode ser configurada em qualquer uma das 5 linguagens
  - Ramificação: permite gerar divergência e convergência de seqüências do programa

SFC - *Sequential Function Chart*

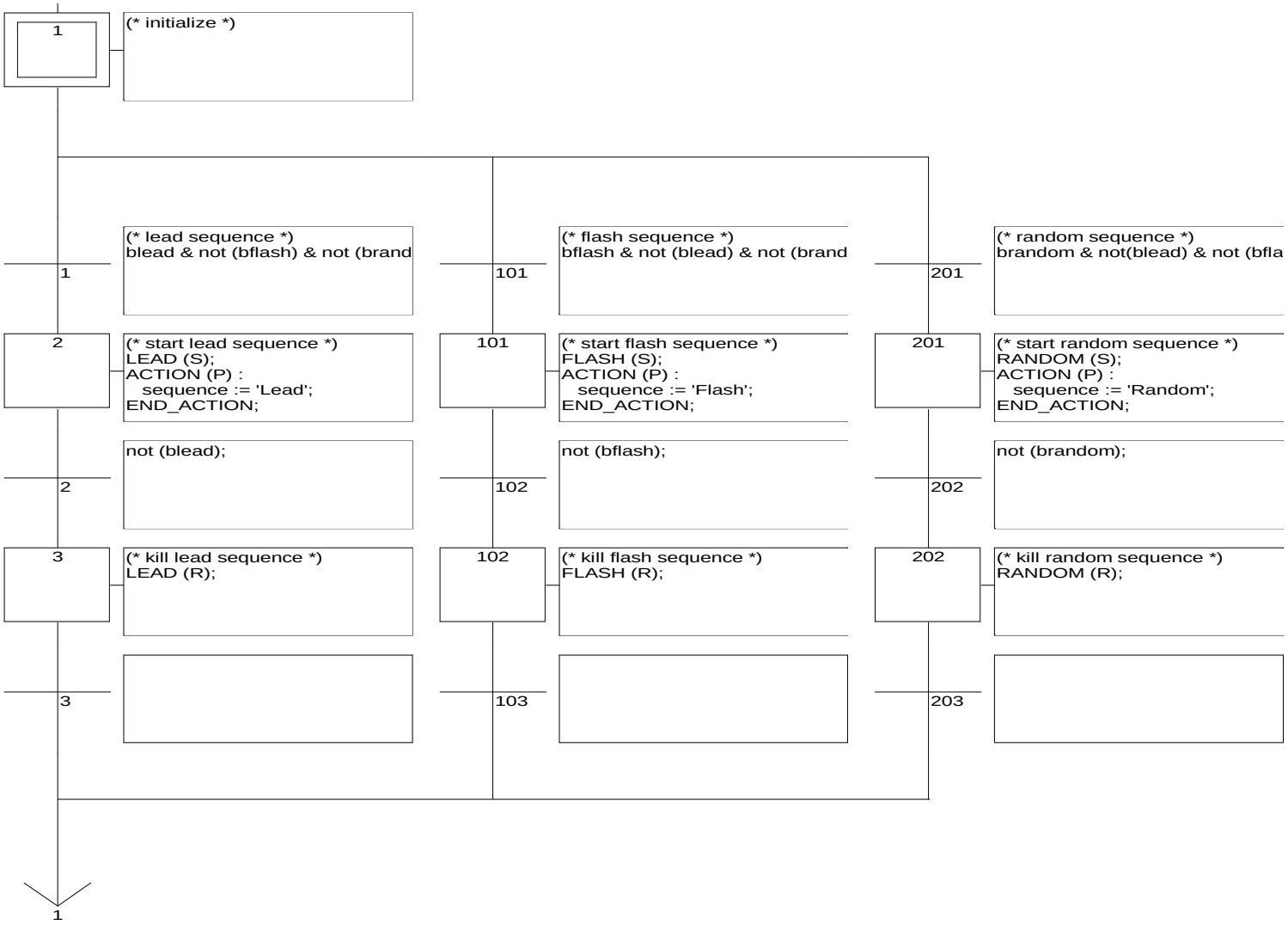


SFC - *Sequential Function Chart*

DECLARAÇÃO	SÍMBOLO	CONTEÚDO DA FUNÇÃO
PASSO		NOME CONDIÇÃO (ATIVA OU INATIVA) TEMPO DECORRIDO
TRANSIÇÃO		NOME, ENTRADA LÓGICA NO DIAGRAMA LADDER OU NOS BLOCOS-DE-FUNÇÕES
AÇÃO		a: QUALIFICADOR (OPCIONAL) b: AÇÃO* c: RETORNO (RESULTADO LÓGICO OPCIONAL)
RAMIFICAÇÕES		DIVERGENTE E SEPARADO (OR)
		DIVERGENTE E SIMULTÂNEO (AND)
		CONVERGENTE E SEPARADO (OR)
		CONVERGENTE E SIMULTÂNEO (AND)

	NENHUM
N	NÃO ARMAZENADO
R	RESET
S	SET (ARMAZENADO)
L	LIMITADO PELO TEMPO
D	COM ATRASO DE TEMPO
P	PULSO
SD	ARMAZENADO E ATRASADO
DS	ATRASADO E ARMAZENADO
SL	ARMAZENADO E LIMITADO

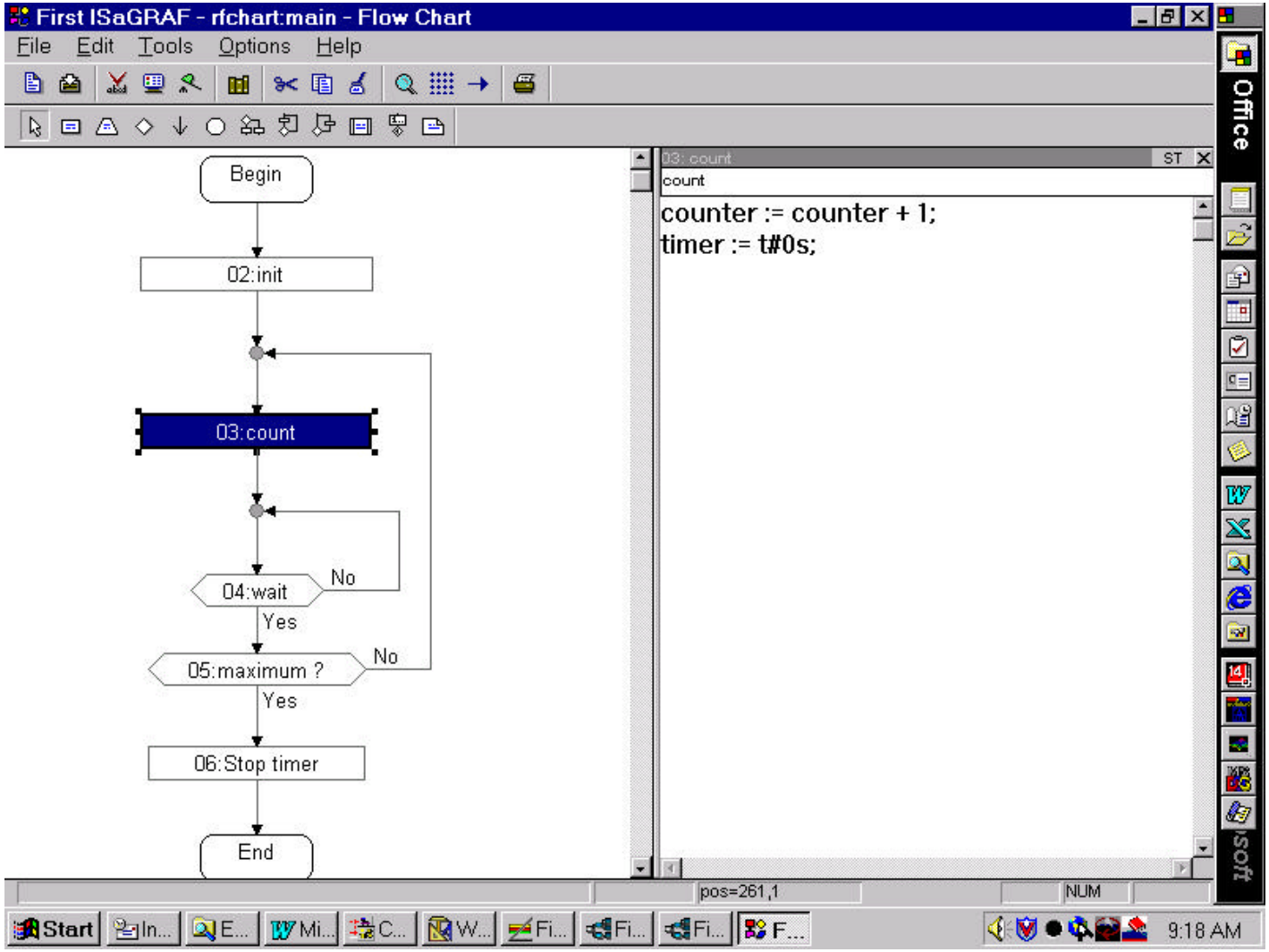
SFC - *Sequential Function Chart* – Exemplo IsaGraf



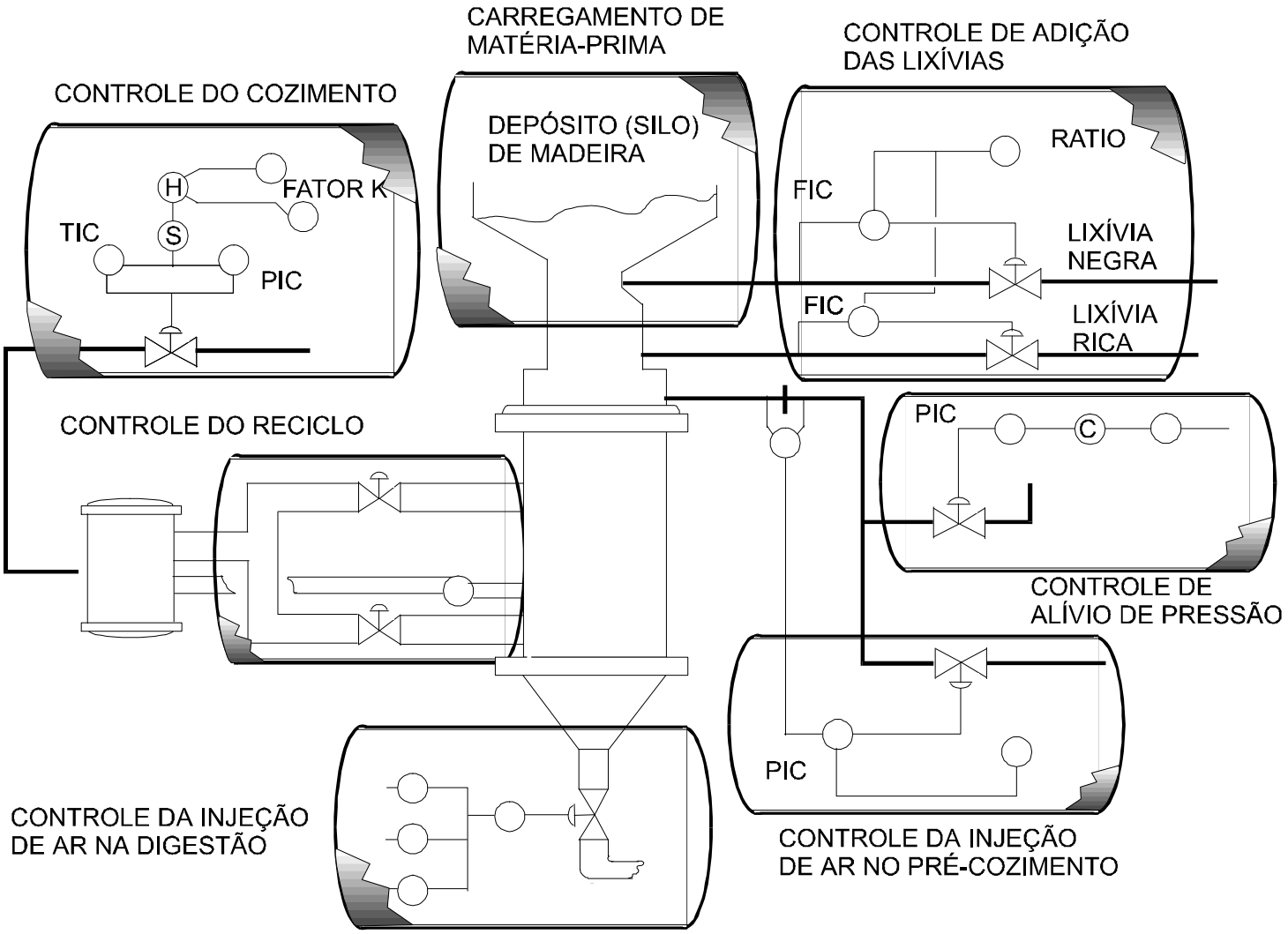
Outras Linguagens - *Flow Chart*, C, *Visual Basic*, Etc…

- A norma permite a utilização de linguagens adicionais para declaração de funções ou blocos de função, devendo obedecer à mesma forma de chamada e troca de dados
- Utilização básica
  - Codificação de algoritmos especiais/complexos
  - Proteção de código proprietário
  - Recursos de programação avançados
  - Bibliotecas dedicadas
- *Flow Chart Programming* poderá ser acrescentada à norma como linguagem padrão

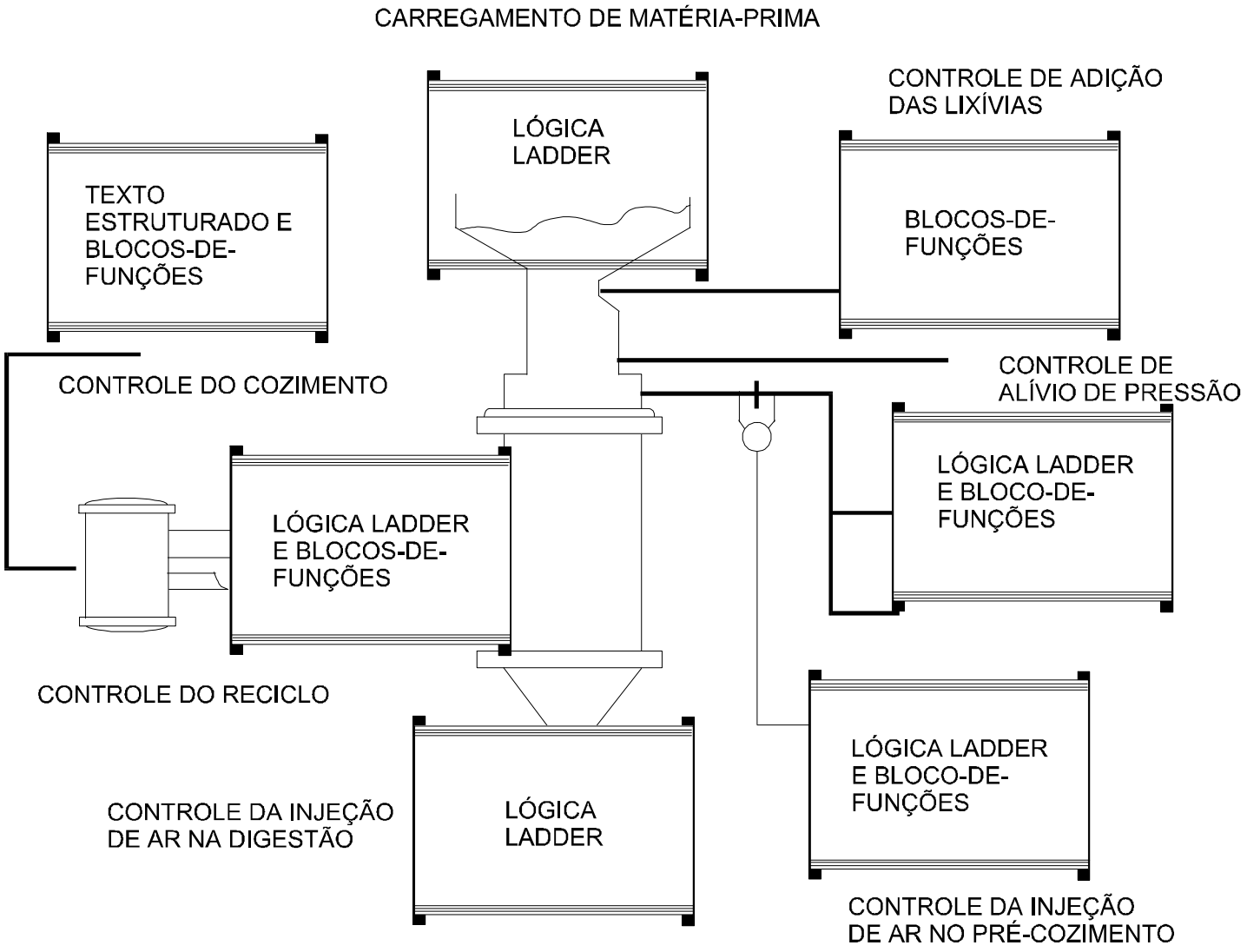
Flow Chart Programming – Exemplo IsaGraf



Análise do problema ® Usar metodologia da norma ISA SP 88

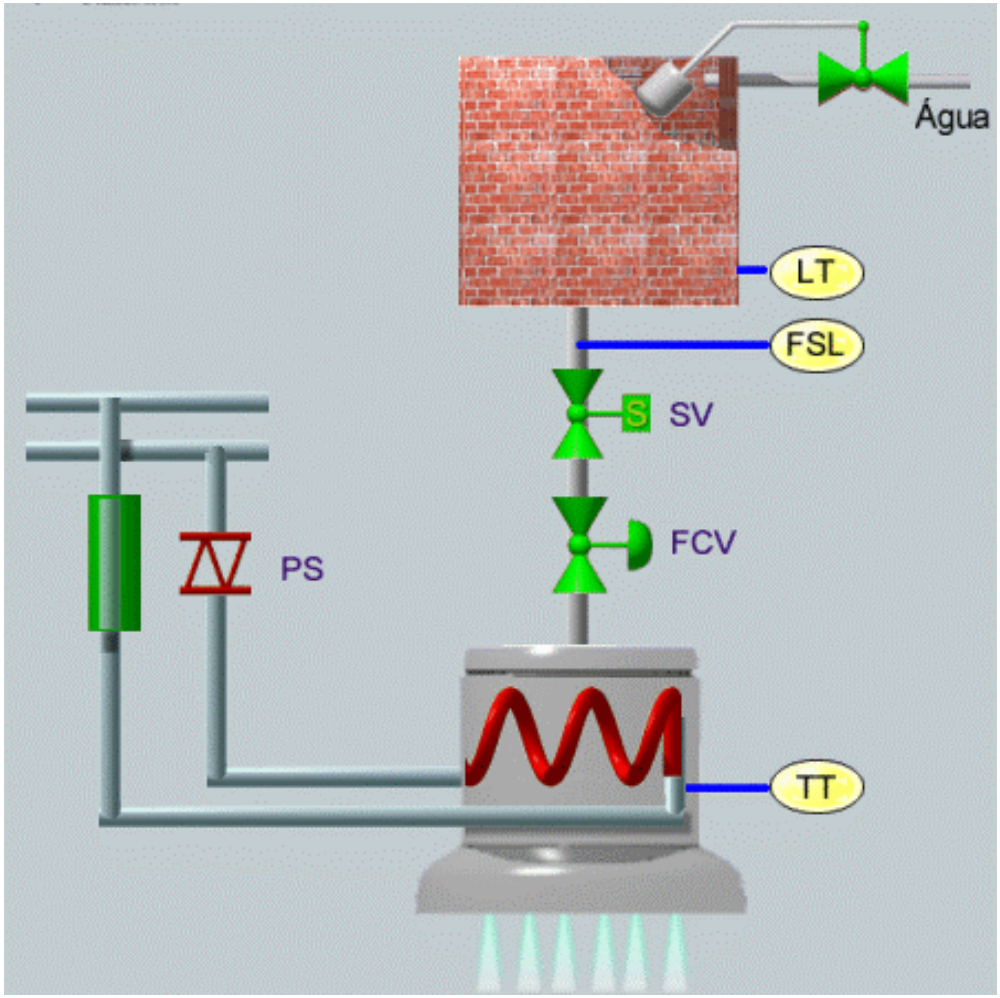


Análise do problema



Exercício

Exercício: Implementar no IsaGraf a lógica de controle de um chuveiro elétrico inteligente



Exercício

Lista de variáveis do Processo

Digitais

- FS_Vazao_Baixa: Entrada da Chave de Fluxo
- SV_Valv_Agua: Saída para comando da Válvula solenóide

Analógicas

- LT_Nivel_Caixa: Entrada do Nível da Caixa D'água
- TT_Temp_Agua: Entrada da Temperatura da Água
- FCV_Vazao_Agua: Saída para controle da Vazão de Água
- PS_Chaveador: Saída para controle do Chaveador

Exercício

Lista de variáveis para comando do Operador

Digitais

- **ON_OFF: Entrada da chave principal**
- **Ligado: Saída para indicação do sistema funcionando**

Analógicas

- **SP_Vazao_Agua: Valor desejado da vazão de saída**
- **SP_Temperatura: Valor desejado da temperatura da água**

Exercício

Considerações

- **Todas as variáveis analógicas são do tipo inteiro, na faixa de 0 a 100**
- **Se o nível da caixa d'água reduzir a zero, a válvula solenóide SV deve ser desligada até que o nível volte à aumentar (restabelecimento)**
- **O Controlador PID poderá ser utilizado conforme exemplo existente no programa *Controle***
- **O chuveiro consegue trabalhar na faixa de 25 a 50 ºC para uma vazão de saída de 0 a 90 %**

Exercício

Objetivo

Desenvolver a lógica de controle do chuveiro de forma a atender às seguintes condições

- **Fazer o controle das funções do chuveiro conforme comandos do operador**
- **Garantir o funcionamento do chuveiro dentro de condições seguras**
- **Proteger o chuveiro contra**
  - **Falta de água**
  - **Elevação da temperatura acima de 65 º C**
- **Estabelecer a seqüência de ligamento e desligamento seguro do chuveiro**

Dúvidas ?