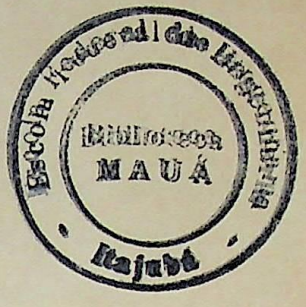
TESE
586

ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ

Eletrônica / Pan fut
621.38:681.31

DISSERTAÇÃO DE MESTRADO



DESENVOLVIMENTO DE UM SIMULADOR DE CONTROLADOR LÓGICO PROGRAMÁVEL EM MICROCOMPUTADORES

CARLOS ALBERTO MURARI PINHEIRO

ORIENTADOR: PROF. JOSÉ ABEL ROYO DOS SANTOS

ITAJUBÁ/MG

1988

CDD. 621.38:681.31 (043.2)
P654 d
TOMBO 586

RESUMO

Este trabalho consiste na implementação de um simulador de Controladores Lógicos Programáveis(CLPs),com microcomputadores compatíveis com a linha IBM PC-XT.

O motivo para tanto, é o fato do CLP, ser um dos tipos de equipamentos digitais de controle, de aplicações gerais em automação de processos industriais. Nas disciplinas de controle em Universidades e Escolas Técnicas, esse equipamento e outros (Computadores de Processo, SDCD, Comandos Numéricos, etc) devem ser vistos.

Entretanto o preço desses equipamentos, mesmos os de pequeno porte (estão na faixa de 500 OTNs), torna-os pouco atrativos, para demonstrações em laboratório, em relação as necessidades de outros equipamentos para disciplinas básicas. Mas como essas instituições, em geral , possuem microcomputadores que são utilizados em diversos cursos, pesquisas, administraçao, etc, pensou-se no uso dos mesmos como simuladores de CLP, para ilustração de aulas teóricas nos cursos de automação.A partir das equações lógicas de um controle seqüencial gera-se na tela ou impressora de um micro, o diagrama de relé equivalente ao processo, ou com um interface de baixo custo, faz-se o controle em tempo real de montagens em bancada ou de pequenos processos.

Com esse propósito foi desenvolvido este trabalho de dissertação, visando melhorar as condições didáticas em disciplinas de controle nos cursos Técnicos e de Engenharia.

ABSTRACT

This work describes the implementation of a Programmable Logical Controller (PLC) simulator for microcomputers IBM PC-XT compatibles.

PLC´s are a type of equipment widely used in automation and industrial process control. The expensive price of this equipment avoid the extensive use of it for training courses in our universities and thechnical schools.

The basic idea of this work is, using availables microcomputers, create a environment to simulate the PLCs. Therefore, given a process, the microcomputer generales a screen based in the logic equations of the sequencial control where, the equivalent relay diagram is displayed and the logic equations can operate, through low cost interface, a set-up a real time system built in laboratory for simulations purposes.

So, the main objective of this work is improve the teaching conditions of the process control by PLC´s.

CONTEÚDO

	Pág.
RESUMO	I
ABSTRACT	II
CONTEÚDO	III
CAPÍTULO 1 - INTRODUÇÃO	01
CAPÍTULO 2 - FORMULAÇÃO DO PROBLEMA	05
CAPÍTULO 3 - INTERFACE PARA SIMULAÇÃO EM TEMPO REAL	10
CAPITULO 4 - ESTRUTURA DO PROGRAMA DE SIMULAÇÃO	15
CAPITULO 5 - EXEMPLOS DE APLICAÇÃO	24
CAPÍTULO 6 - COMENTÁRIOS E CONCLUSÕES	39
APÊNDICE A - MANUAL DE UTILIZAÇÃO DO SIMULADOR	45
APÊNDICE B - LISTAGEM DO PROGRAMA SIMULADOR	51
REFERÊNCIAS BIBLIOGRÁFICAS	77

CAPÍTULO 1

INTRODUÇÃO

Os Controladores Lógicos Programáveis(CLPs) ou Controladores Programáveis(CPs), foram desenvolvidos nos anos 50/60 com a finalidade de substituir painéis de relés, em controles baseados em lógica combinacional /seqüencial, em linhas de montagem nas indústrias de manufatura(principalmente automobilística), sendo progressivamente adotados pelas indústrias de processos(siderúrgicas,petroquímicas,celulose, etc).

Além das funções usuais básicas do tipo "E", "OU", "Negação", "Memorização" e "Temporização", os CLPs mais recentes possuem outras instruções como "Contagem", "Manipulação de bits", "Malhas PID", etc. *

Junto com os Computadores de Processo e os SDCDs(Sistemas Digitais de Controles Distribuído), os CLPs, são os três tipos de equipamentos digitais de controle de aplicações gerais, mais utilizados na automação industrial (os Comandos Numéricos Computadorizados-CNC, são mais dedicados a máquinas ferramentas).

A lógica em controles seqüenciais, sem a utilização de dispositivos eletrônicos, é implementada por contatos eletromecânicos com a utilização de relés e contatores. Contatos em série representam funções "E", em paralelo "OU", etc. Como desvantagens pode-se citar :

- fadiga e desgaste dos contatos eletromecânicos;
- velocidade limitada,dezenas de milisegundos;
- gabinetes e painéis grandes;
- falta de flexibilidade em alterações de lógica (programação), implicando em modificações de circuitos e fiações.

A utilização de circuitos eletrônicos, implementando funções lógicas , trouxe grandes melhorias nos problemas de :

* [1],[2],[3],[4]

desgaste; tempo de chaveamento; tamanho e consumo. O advento das unidades de processamento ou processador, permitiu o conceito de programação a esses equipamentos. As alterações em programas(software), não implicam mais em modificações nos circuitos e fiações(hardware) , mas sim na mudança de dados contidos em elementos de armazenamento(memórias) , obtendo-se flexibilidade em modificações na lógica de um controle.

Os CLPs possuem a seguinte estrutura básica :-

- Processador : Onde são realizadas as funções lógicas/ aritméticas.
- Bancos de Memória : Onde são armazenados os programas básicos, aplicativos e dados a serem executados.
- Interfaces : Onde entram ou saem as informações ou comandos para o processo controlado.
- Periféricos : De comunicação, teclado/display, vídeo, disco, impressora, e outros.

A figura 1 mostra um diagrama em blocos da estrutura básica de um Controlador Programável.

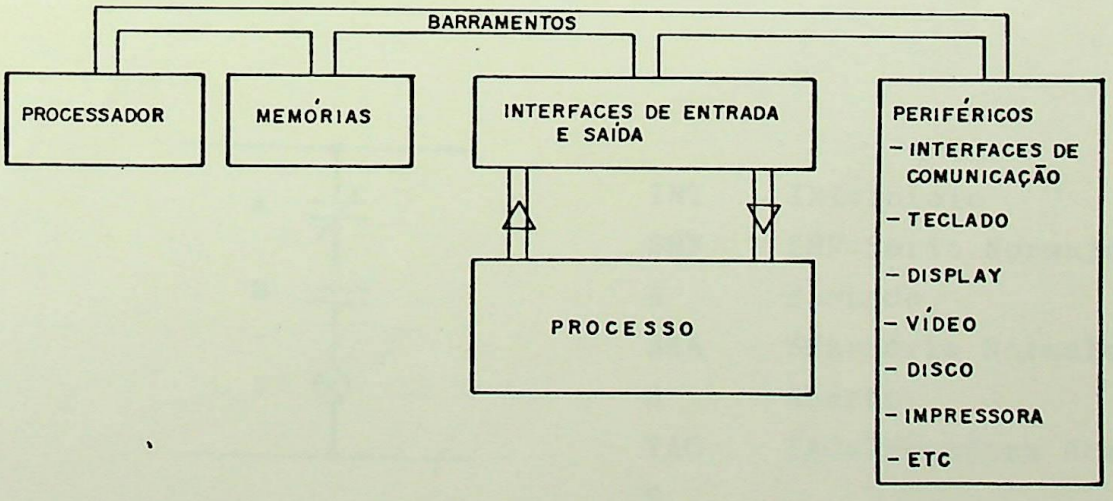
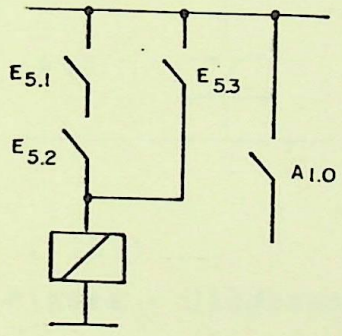


Figura 1 - Estrutra básica de um CLP.

Existem várias linguagens de programação para CLPs, adotadas pelos fabricantes :

- Diagrama de contatos.
- Lista de instruções.
- Diagramas funcionais.
- Etc.

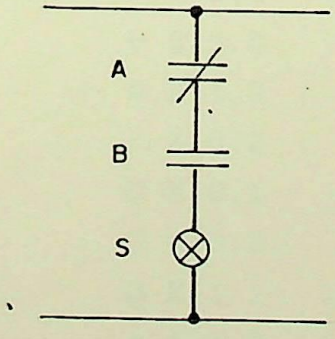
As mais comuns são listas de instruções e diagrama de contatos. O mercado tem um bom número de fabricantes , entre eles : Altus, Digicon, Maxitec, Metal Leve, Pulse, Villares, WEG, etc (alguns associados a multinacionais: Allen-Bradley, Fuji, Siemens,e outras). Para ilustrar vamos mostrar alguns trechos de programas para CLPs comerciais na figura 2. *



(a) Diagrama de contatos

```
U E 5.1        E=Entrada XY
U E 5.2        U=Função E
O E 5.3        O=Função OU
A 1.0          A=Saída XY
```

(b) Lista de Instruções



(c) Diagrama de contatos

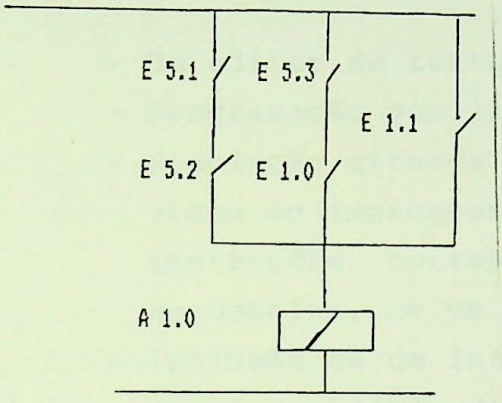
```
INI     INI=Início
SNF     SNF=Série Normalmente
A       Fechado
SNA     SNA=Série Normalmente
B       Aberto
TAC     TAC=Transfira Acumulador
S
```

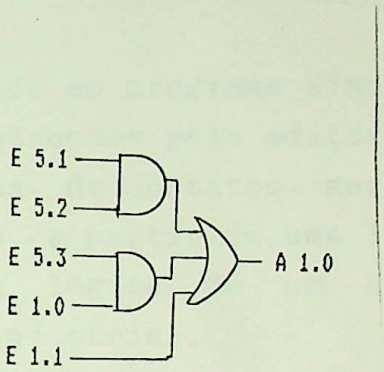
(d) Lista de Instruções

Figura 2 - Exemplos de programas para CLPs comerciais.

* [8]

A grande flexiblidade dos CLPs em relaçao aos painéis de relés, está na facilidade em modificações de lógica, por exemplo, seja a mudança no diagrama da figura 2a, para as 2e, 2f e 2g respectivamente. Nota-se que basta alterar a lista de instruções(software) do programa(figura 2g) de controle, ao passo que em dispositivos a relé ou portas semicondutoras, as modificações são em fiações ou portas(hardware),ver figuras 2e e 2f.





(2e) (2f)

Figura - Diagramas de Contato e Funcional.

```
U E 5.1
U E 5.2
= M 0.1          M = Memorização
U E 5.3
U E 1.0
= M 0.2
O M 0.1
O M 0.2
O E 1.1
= A 1.0
```

Figura 2g - Lista de instruções modificada.

CAPÍTULO 2

FORMULAÇÃO DO PROBLEMA

Propôs-se nessa dissertação, o desenvolvimento de um simulador de CLP , para microcomputadores com as seguintes características :

- Um editor de texto incorporado ao programa simulador.
- Programação por lista de instruções pelo editor.
- Simulação através de diagrama de contatos, gerados no vídeo ou impressora do micro, a partir de uma lista de instruções correspondente a lógica de um controle seqüencial, de um processo industrial.
- Inclusão de um interface de baixo custo, para controle em tempo real de montagens de laboratório , ou de pequenos processos experimentais.
- Funções do tipo "."(E), "+"(OU), "N"(Negação) , "T"(Temporização) e "V"(Variáveis auxiliares).
- Capacidade de : Entradas(de E11 a E88) ; Saídas (de S11 a S18); Temporizadores(de T11 a T88) de 00 a 99 segundos com variações de 1s; Variáveis(de V11 a V88).
- Total de 200 linhas por programa editado,possibilidade de comentários para documentação dos programas.
- Conceito de programação estruturada , onde a lista de instruções ou diagrama de contatos equivalente, possua apenas as associações básicas das funções de programação ou de representação.

Por motivos de estrutura e apresentação, fixou-se a montagem das listas de instruções em torno de regras (sintaxe) que serão apresentadas a seguir :

REGRAS DE SINTAXE

- Toda a entrada, saída, variável auxiliar e temporizador é representado por uma letra maiúscula E, S, V, T respectivamente, seguidos por dois algarísmos de 1 a 8 (p.ex. E11, S18, V81, T88), onde para as entradas e saídas do interface o primeiro número representa o conector ou endereço das portas e o segundo as linhas das mesmas.

- A negação do estado de uma entrada, saída, variável e temporizador é obtida com a inclusão de N após os dois números de identificação(p.ex. E11N, S13N, V81N, T22N).

- A função "E" é implementada por um "." , exemplo: E11.E12; E12N.E32.E71;etc.

- A função "OU" é programada com "+" e "( )", exemplo: (E11+E12); (E32N+E41N);etc. Por motivos de estruturação não é permitido mais de duas associações "OU" por parênteses, para isto utiliza-se as variáveis auxiliares.

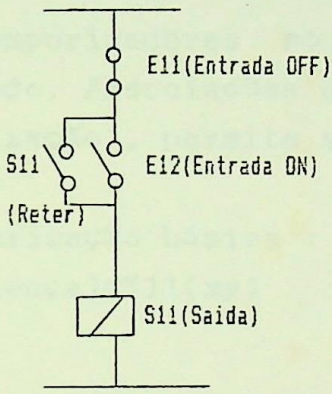
- As atribuições de resultados são sempre realizadas no final da expressão, com o símbolo "=" , exemplo : (E11+E12)=S11; E11.E12=S12; seja adaptar a lógica S13=E11+E12+E13 para as regras em questão, tem-se: (E11+E12)=V11 e (V11+E13)=S13.

- Pode ter-se até seis funções em série, por linha de instrução, exemplo : E11.E12N.(E12+E13N).(E11N+E18).E15.E17N=S18. Para equações maiores usa-se as variáveis auxiliares.

- Não se deve deixar nenhum espaço em branco do início ao fim da expressão, a não ser depois do último caracter da mesma. Comentários podem adicionados após o finalizador espaço em branco ou ponto-e-vírgula(";") no fim da sentença.

- Os valores de programação dos temporizadores, deve estar entre parênteses e terem sempre dois dígitos de 00 a 99(s) p.ex. E11.E12=T12(06); (E12+T11N)=T13(57); etc.

- Com associações de funções "E", "OU", "Negação" pode-se obter "Memorizações" e outros tipos de lógica. Exemplo : Seja montar a lista de instrução dos diagramas de retenção(memorização) a seguir :-



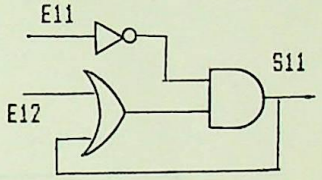


Diagrama de Contato

Diagrama Funcional

A lista equivalente é : E11N.(E12+S11)=S11 .

- Não se pode atribuir um resultado apenas a uma entrada, variável, temporizador e saída, sem uma operação lógica, por exemplo : E11=S11 {Errado} ; E11.E11=S11 {Certo}.

- Toda programa deve acabar com a letra F na última linha :

```
34 E16.E17=S14 ; Fim da lista.
35 F
```

- Qualquer lista de instruções que não obedeçam essas regras, quando executadas, terão mensagens de erro emitidas pelo simulador.

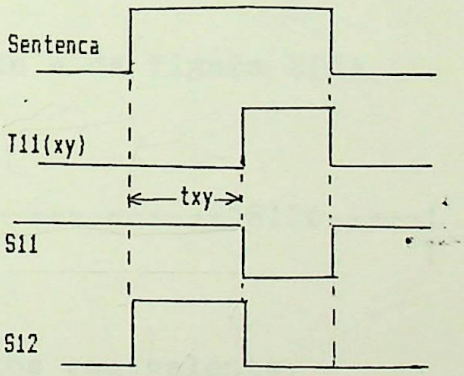
- Nos diagramas gerados nas simulações, os contatos são representados da seguinte forma :

-----] [----- = Aberto.
-----[ ]----- = Fechado.
-----) (----- = Desativado ou não operado.
-----( )----- = Ativado ou operado.

- Os temporizadores só operam quando decorrido o tempo programado. Associações de operações "E", "OU", "Negação" e "Temporização", permite várias formas de atrasos(delays) :-

a) Temporização básica :
   [Sentença]=T11(xy)

b) Básica invertida :
   [Sentença]=T11(xy) ==>
   T11N.T11N=S11



c) Monoestável
   [Sentença]=T11(xy)
   [Sentença].T11N=S12

d) Combinações dos tipos anteriores, permite outras formas de temporizações.

Seja converter os programas da figura 2 com as definições formuladas : -

- Da figura 2(b), fazendo A 1.0 = S11, tem-se:

```
1 E51.E52=V11 ;lista de instruções equivalente a da fig.
2 (V11+E53)=S11 ;2(b) utilizando o simulador.
```

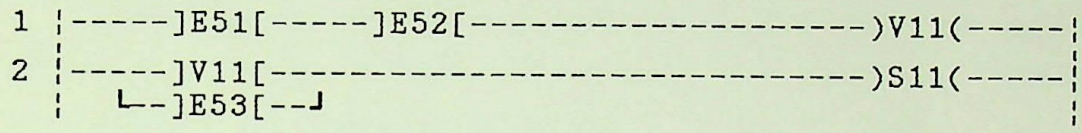


Figura 3 - Diagrama de contatos equivalente gerado.

- Da figura 2(d), fazendo A=E11, B=E12, S=S12, tem-se:

```
3 E11N.E12=S12 ;lista equivalente a da figura 2(d)
```

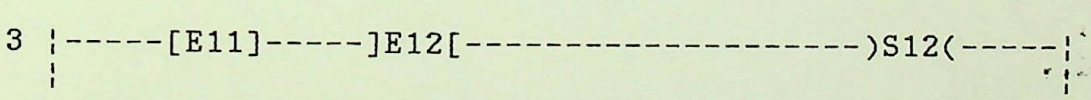


Figura 4 - Diagrama de contatos equivalente.

Exemplos detalhados de aplicação, serão vistos no capítulo 5.

CAPÍTULO 3

INTERFACE PARA SIMULAÇÃO EM TEMPO REAL

Um dos objetivos desse trabalho, além da simulação de diagrama de contatos de um controle seqüencial, em vídeo ou impressora, é a possibilidade de se acionar montagens experimentais ou maquetes em escala reduzida, análogo a um processo industrial.

Para tanto, necessita-se de um interface de entrada e saída, que seja acoplado ao microcomputador. Procurou-se desenvolver um circuito simples e de baixo custo , utilizando componentes fáceis de serem encontrados no mercado nacional. O diagrama desse circuito está na figura 5a , consta de : um circuito lógico que combina as linhas de endereçamento e controle do microcomputador, para habilitar portas de entrada (compostas de buffers) e portas de saída (implementadas por flip-flops), de modo a ter-se um interface entre os barramentos de dados do microcumputador e sensores/atuadores de um processo.

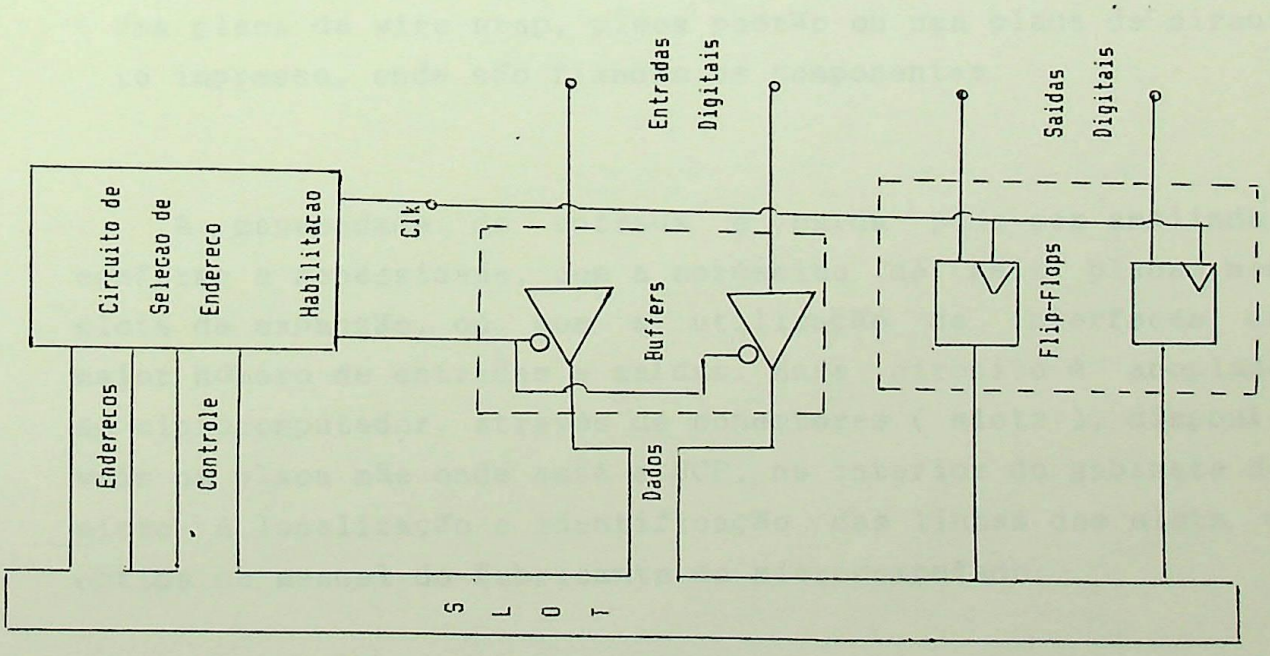


Figura 5a - Diagrama do Interface.

O esquema da figura 5b é a implementação prática do interface, composto por :

- Um circuito de seleção de endereços, composto pelas portas OU (74LS32), pelo decodificador (74LS138) e por um comparador (74LS85), onde são selecionados os endereços vagos para expansões no microcomputador.

- Uma porta de entrada de 8 bits(74LS244), para leitura de informaçãoes de um processo(sensores, estado de relés, finais-de-curso, etc).

- Uma porta de saída de 8 bits(74LS273), para comandar atuadores de um controle(contatores, servoválvulas,triacs, reles,motores e outros).

- Isoladores (ver figura 6b), nas entradas e saídas, para proteger as portas e o própio barramento dos conectores de expansão(slots) do micro, contra eventuais distúrbios, sobrecargas e sobretensões externas.

- Uma placa de wire-wrap, placa padrão ou uma placa de circuito impresso, onde são fixados os componentes.

A capacidade de entrada e saída pode ser ampliada, conforme a necessidade, com a acréscimo de mais placas nos slots de expansão, ou com a utilização de interfaces de maior número de entradas e saídas. Esse circuito é acoplado ao microcomputador, através de conectores (slots), disponíveis na placa mãe onde está a UCP, no interior do gabinete do micro.. A localização e identificação das linhas dos slots é obtida no manual do fabricante do microcomputador.

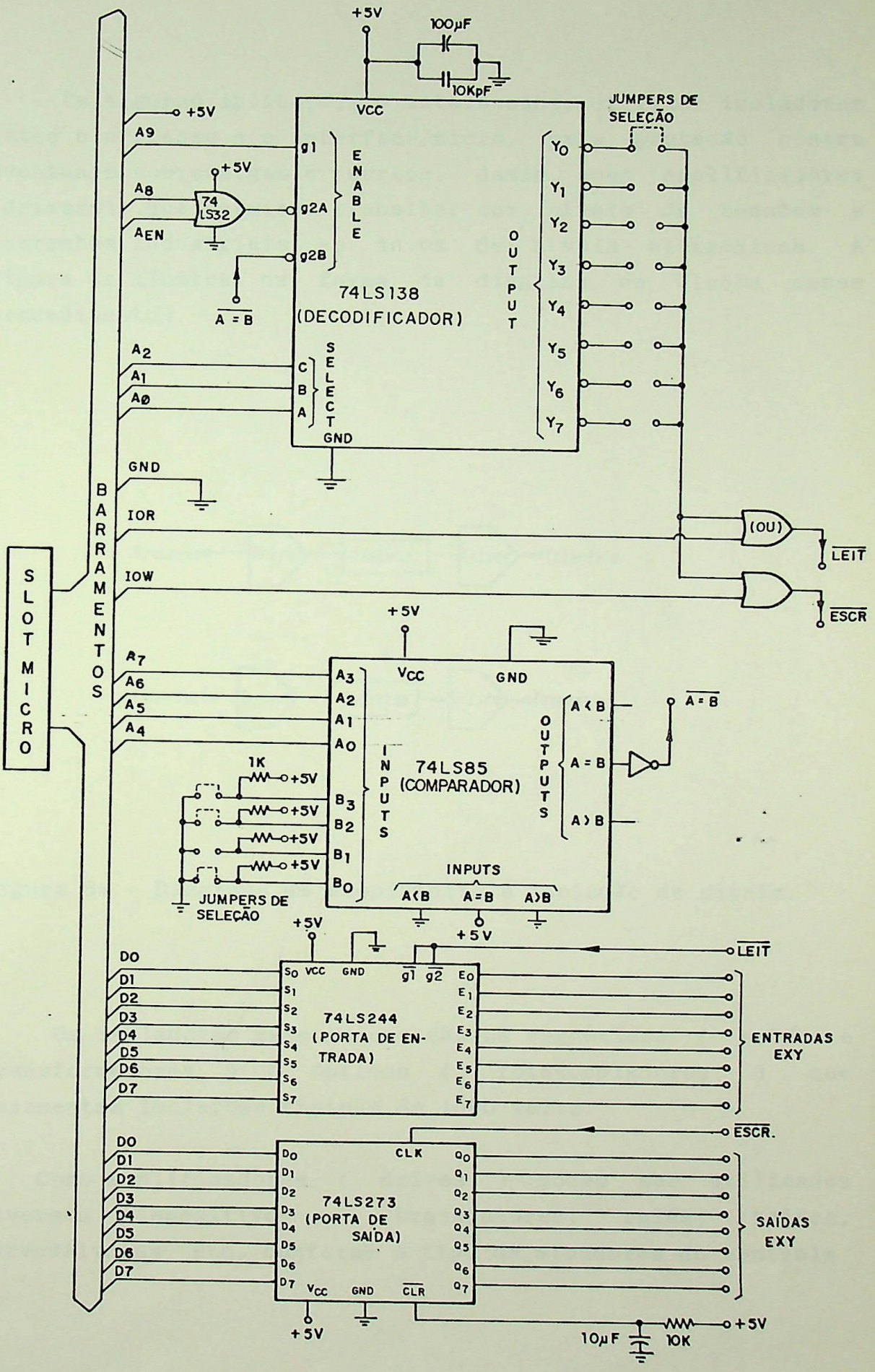


Figura 5b - Interface de entrada e saída digital.

Em algumas aplicações é interessante utilizar isoladores entre o processo e o interface/micro, para proteção contra eventuais sobrecargas e surtos. Assim como amplificadores (drivers), que permitam trabalhar com níveis de tensões e correntes industriais, ao invés de níveis eletrônicos. A figura 6a ilustra na forma de diagrama de blocos esses procedimentos.

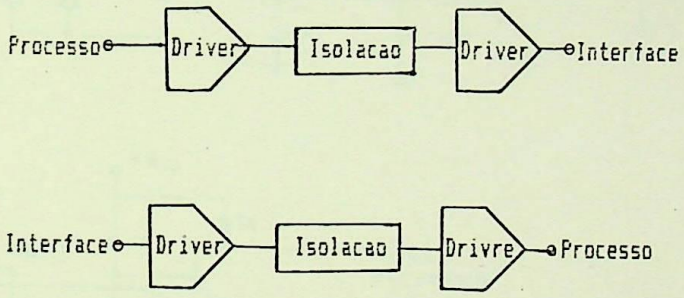


Figura 6a - Diagrama de acoplamento e isolação de sinais.

Os isoladores mais comuns são os magnéticos (relés e transformadores) e ópticos (fotoacopladores) que apresentam isolações típicas de 1500 volts.

Como amplificadores (drives) podem ser utilizados diversos dispositivos : transistores; relés; triacs, servoválvulas, etc, conforme o tipo de atuadores do controle.

Os esquemas da figura 6b, mostram algumas opções de acoplamento das entradas e saídas isoladas, com alguns níveis típicos de sinais provenientes de um controle.

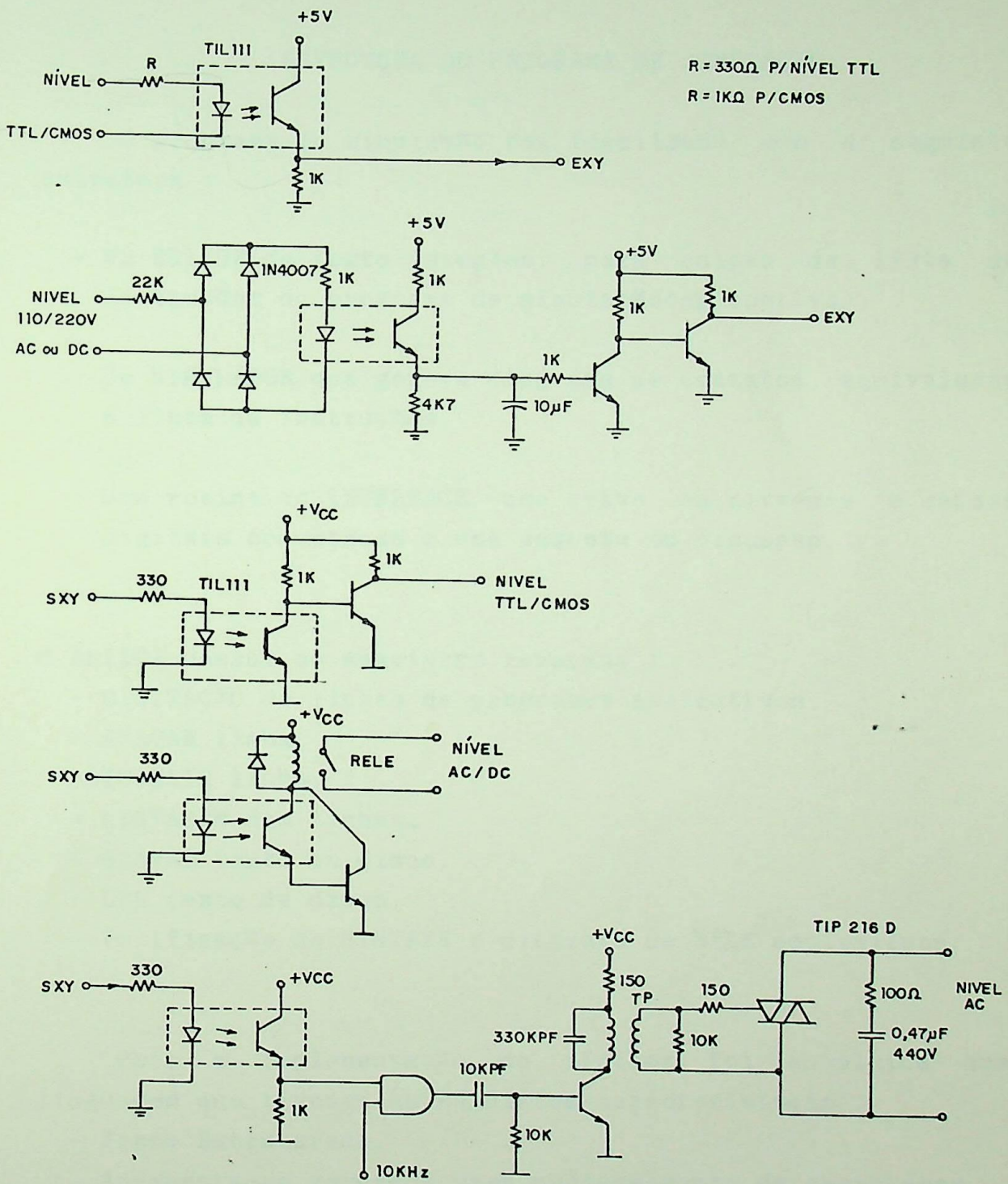


Figura 6b - Acoplamento de sinais das entradas e saídas.

CAPITULO 4

ESTRUTURA DO PROGRAMA DE SIMULAÇÃO

O programa de simulação foi idealizado com a seguinte estrutura :

- Um EDITOR de texto simples, para edição da lista de instruções do programa de simulação(aplicativo).

- Um SIMULADOR que gere o diagrama de contatos equivalente a lista de instruções.

- Uma rotina de INTERFACE que ative as entradas e saídas digitais conectadas a uma maquete ou processo.

O EDITOR possue os seguintes recursos :

- DIGITAÇÃO de linhas de programas aplicativos.
- APAGAR linha.
- INSERIR linha.
- LISTAGEM das linhas.
- GRAVAR texto em disco.
- LER texto de disco.
- Verificação de SINTAXE e diagrama de RELÉ equivalente.

Para a implementação do sistema foi escolhida uma linguagem que tivesse as seguintes características :

- Fosse Estruturada.
- Apresenta-se recursos para processamento de caracteres.
- Instruções de entrada e saída para acionar interfaces.
- Boa velocidade de processamento.

A escolha foi um compilador Pascal e um sistema operacional padrão. Utilizou-se o Turbo Pascal 3.0 da Borland, contido

num único disquete de 5 1/4" e um sistema operacional PC-DOS versão 3.1 da IBM , podendo ser utilizado outros compiladores e sistemas operacionais.

O programa desenvolvido básicamente consta de interpretadores de comandos.A listagem do fonte encontra-se no apêndice B , o programa está devidamente documentado, sendo relativamente fácil a sua leitura e compreensão. Passaremos agora a descrever a funcão das principais rotinas ou subrotinas (procedure) do mesmo :-

- Status : Bloco de leitura das entradas digitais através do teclado do microcomputador.

- OnOrOff : Bloco de teste de estado das entradas, saídas, variáveis e temporizadores.

- Sint_Rele : Interpretador de comandos para verificação de sintaxe e montagem do diagrama de contatos equivalente a uma lista de instruções de um programa aplicativo.

- Tempo : Subrotina de leitura do relógio do micro.

- Logica : Interpretador de comandos para implementação da lógica combinacional/seqüencial da lista de intruções de um programa editado.

- Verif_Sint : Subrotina para verificar erros de sintaxe em programas que são utilizados pela rotina Interface.

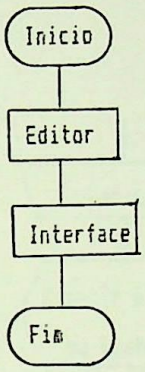
- Editor.

- Simulador.

- Interface.

Vejamos agora a constituição básica das principais rotinas utilizadas no simulador, para o acionamento das portas de entrada e saída digitais, controlando algum sistema em tempo real.

Os blocos básicos empregados são :

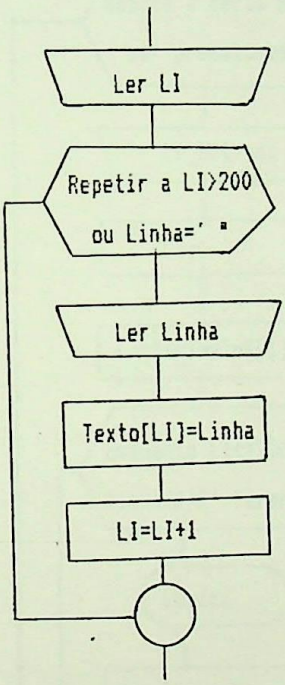


Editor-Digitação

Essa procedure (subrotina) possui a rotina Digitação, onde se faz a entrada, pelo teclado do micro, da lista de instruções de um controle lógico. Para tanto foi utilizado variáveis com as seguintes funções :

LI = Número da linha de instrução atual.
Linha = Variável que recebe os caracteres das instruções.
Texto[LI] = Vetor que armazena as linhas de caracteres.

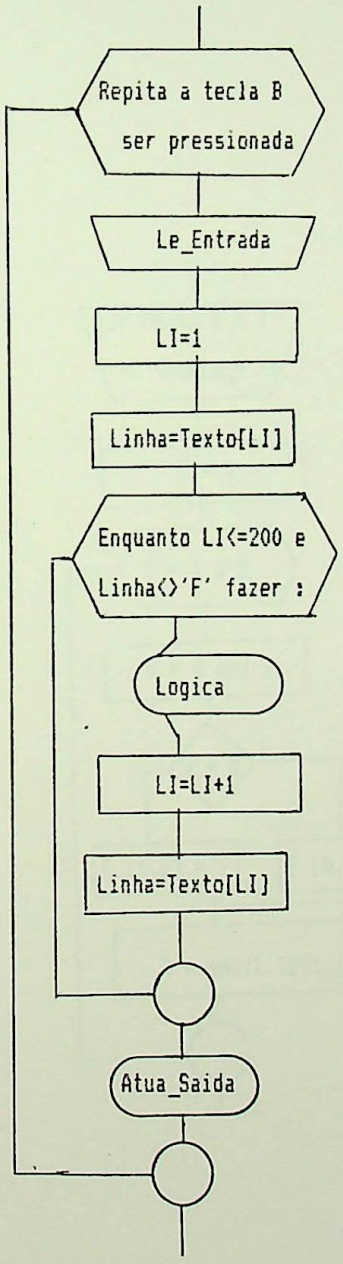
O fluxograma a seguir indica a operação de entrada das listas de instruções que são armazenadas no vetor.



- Fluxograma Digitação -

Interface

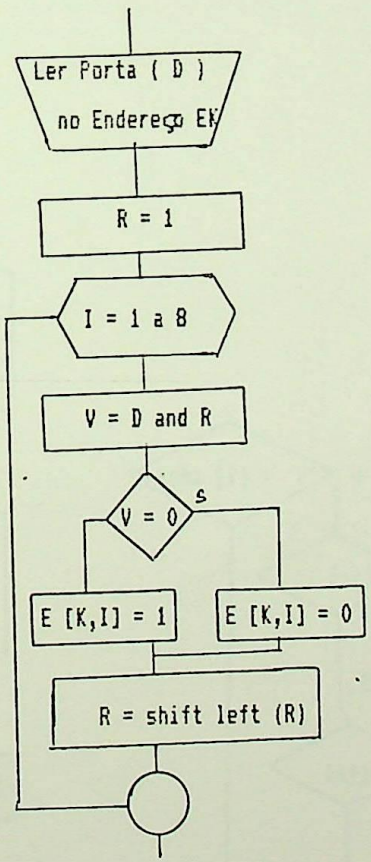
A subrotina Interface chama outras, que fazem a leitura (varredura) das entradas digitais, implementam a lógica de controle a partir da lista de instruções e ativam as saídas do interface.



- Fluxograma Interface -

Le_Entrada

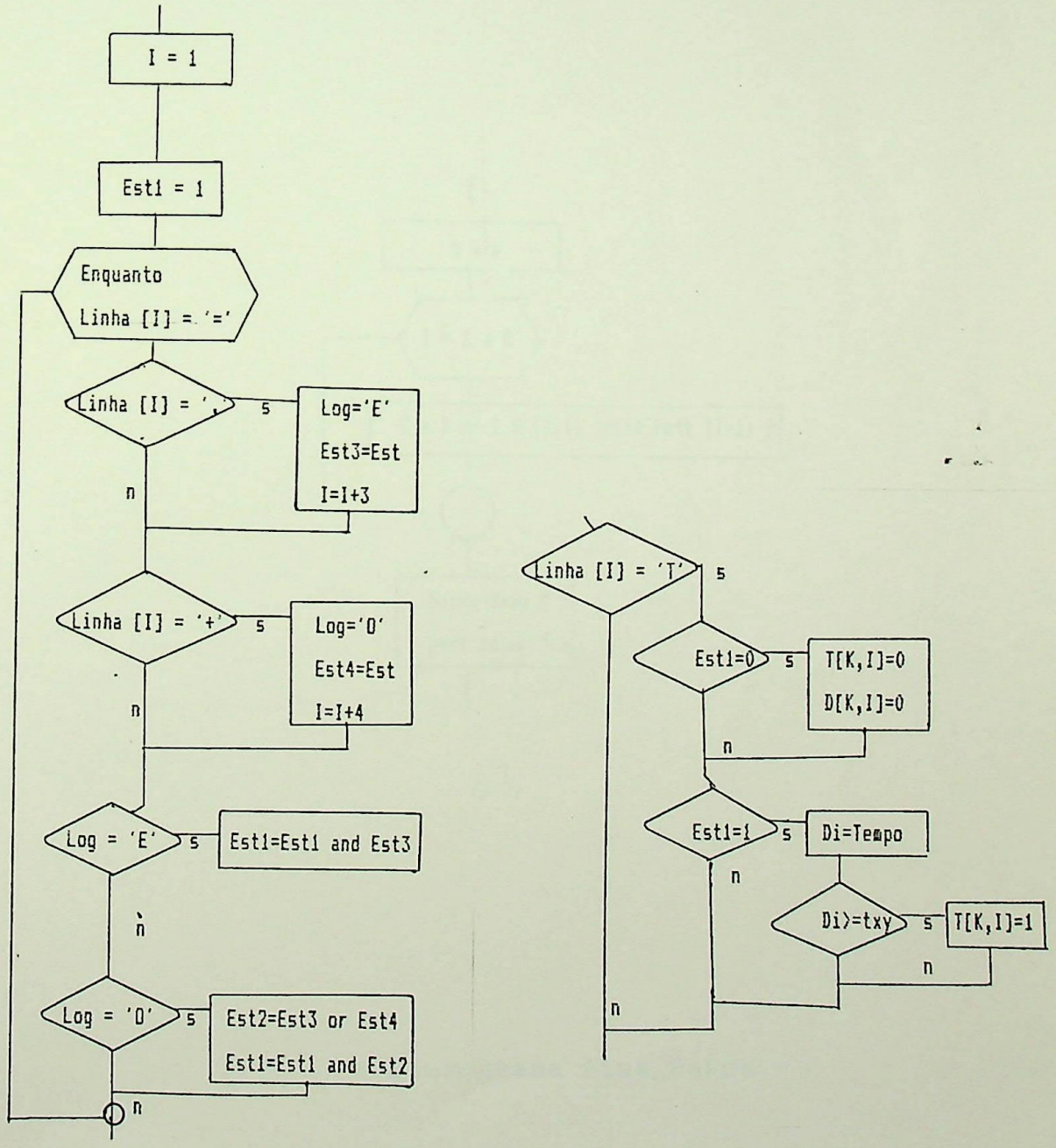
Esta subrotina realiza a leitura dos oito bits de uma porta de entrada e armazena o estado dos mesmos na matriz E[K,I] através de operações de teste e deslocamento de bits.



- Fluxograma Le_Entrada -

Lógica

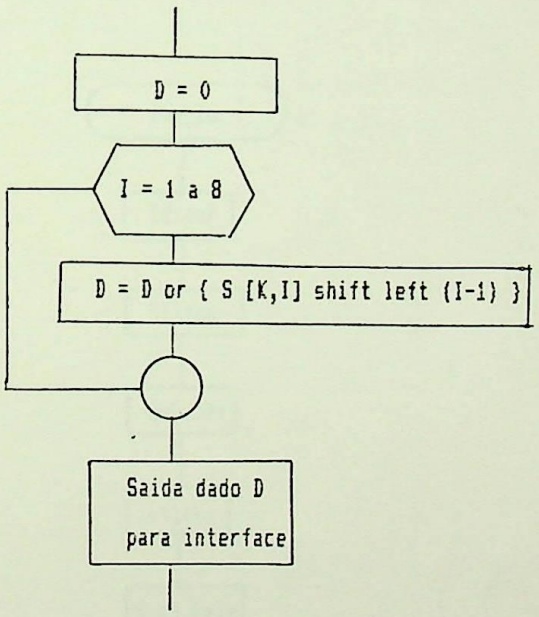
A procedure implementa a lógica da lista de instruções. Básicamente é um interpretador de comandos, que varre os caracteres de uma linha de instrução, realizando as funções "E", "OU", "Negação" e "Temporização", onde :
I = Apontador de caracter; Linha = Linha de instrução;
Log = Variável indicadora de função lógica; Estx = Estado;
Tempo = Subrotina de leitura do relógio do micro.



- Fluxograma Lógica -

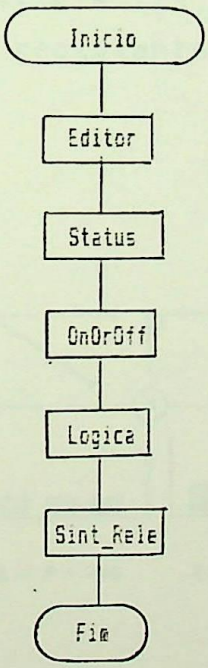
Atua_Saída

Esta subrotina obtem os dados da matriz de saída S[K,I] e os transforma em um byte, acionando a porta de saída.



- Fluxograma Atua_Saída -

Finalmente as partes do programa envolvidos na simulação de diagramas de contato, no vídeo ou impressora , a partir de listas de intruções de um controle, são indicadas no algoritmo abaixo. A subrotina "Sint_Rele" é um interpretador de comandos, que gera no vídeo ou impressora, os diagramas correspondentes.



Comentários a respeito dos tempos de processamento envolvidos e sugestões para incrementar esse simulador serão mencionados no capítulo 6.

CAPÍTULO 5

EXEMPLOS DE APLICAÇÃO

EXEMPLO 1

Um sistema de seleção de volumes por faixa de pesos, é acionado por uma esteira transportadora, na qual existe um sistema de pesagem(sensores), que fornece 4 sinais correspondentes a uma medição (veja figura 7), onde : um indica a presença(energizado) ou não(desenergizado) de volumes (sensor de peças); um outro se o volume está na faixa de 0,5 a 2Kgf (ativado se verdadeiro); outro de 2 a 5Kgf; e finalmente um último sinal para pesos maiores que 5Kgf. Através de 3 atuadores(A, B, C) se desvia a trajetória dos volumes da esteira transportadora, para recipientes conforme a faixa de pesos.

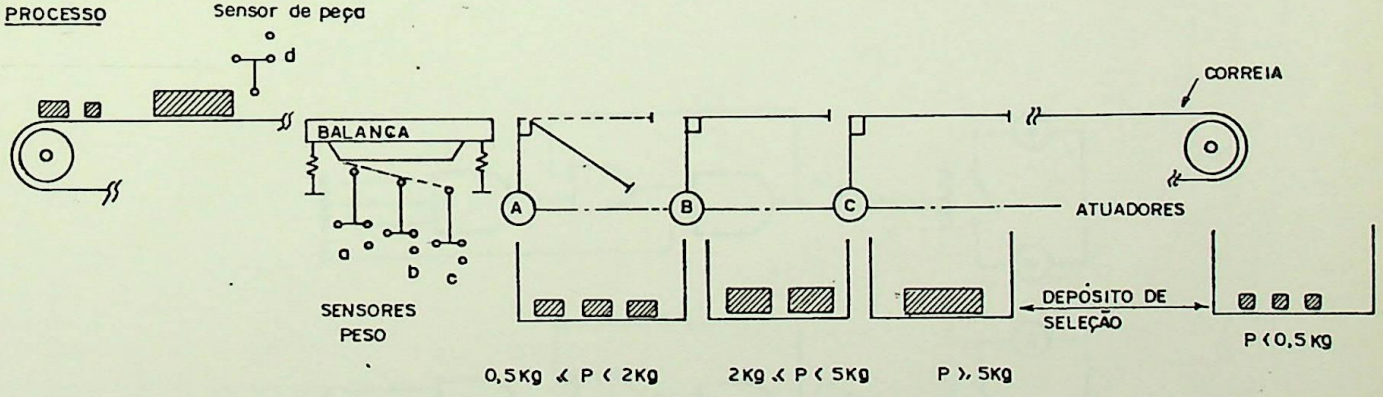


Figura 7 - Desenho do Exemplo 1.

As figuras 8 e 9 representam duas formas de se automatizar esse processo, uma utilizando relés ou contatores e outra com portas lógicas semicondutoras.

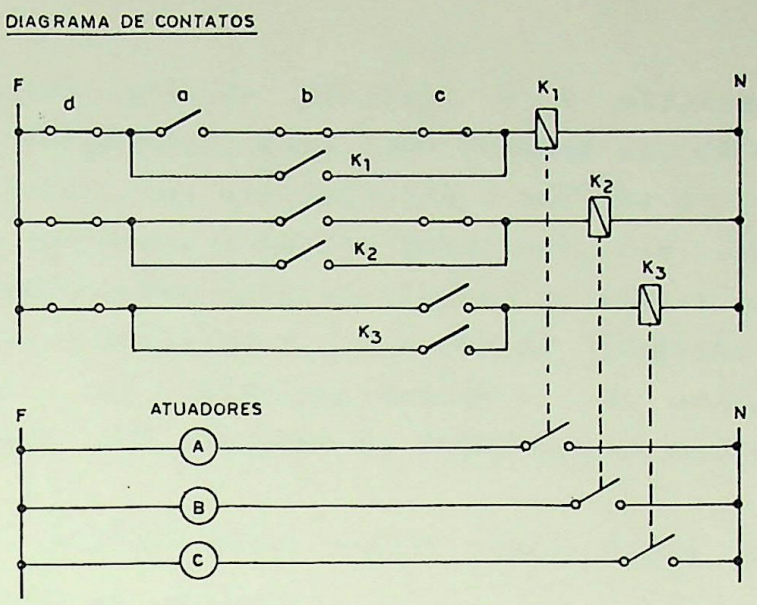
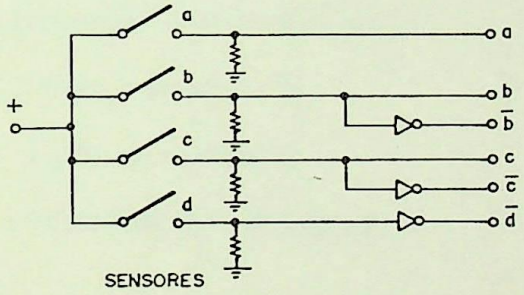


Figura 8 Diagrama de relé do Exemplo 1.



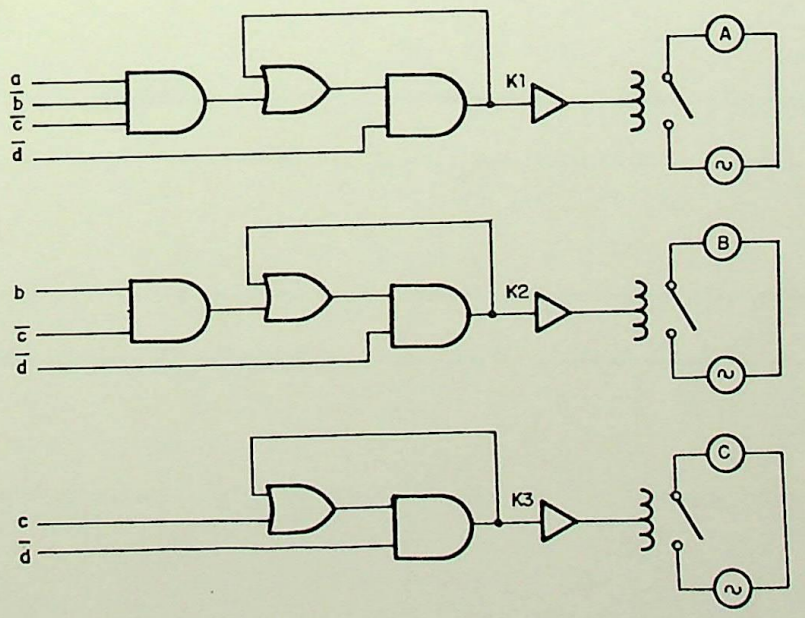


Figura 9 Diagrama funcional com portas lógicas.

Uma outra solução possível é a utilização de um Controlador Programável.Para isso faremos uso do simulador de CLP desenvolvido. Uma vez definida a solução do problema, por diagrama de contatos, diagrama funcional, etc, podemos montar a lista de instruções correspondente, no editor do programa (figura 10) e simularmos na forma de diagrama de contatos (figuras 11) no vídeo/impressora , ou acionar chaves, lâmpadas, leds, etc, através do interface de entrada e saída.

```
1 ; Do Exemplo 1 : a=E11, b=E12, c=E13, d=E14, A=S11, B=S12, C=S13
2 ;Lista de comandos do controle :
3 E11.E12N.E13N=V11 ; Implementando primeira faixa de peso
4 (V11+S11).E14N=S11 ; Selo e sensor de volume para atuador A
5 E12.E13N=V12 ; Idem segunda faixa de peso
6 (V12+S12).E14N=S12 ; Idem atuador B
7 (E13+S13).E14N=S13 ; Idem atuador C
8 ; Final da lista de comandos do Exemplo 1
9 F
```

Figura 10 - Lista de instruções editada.

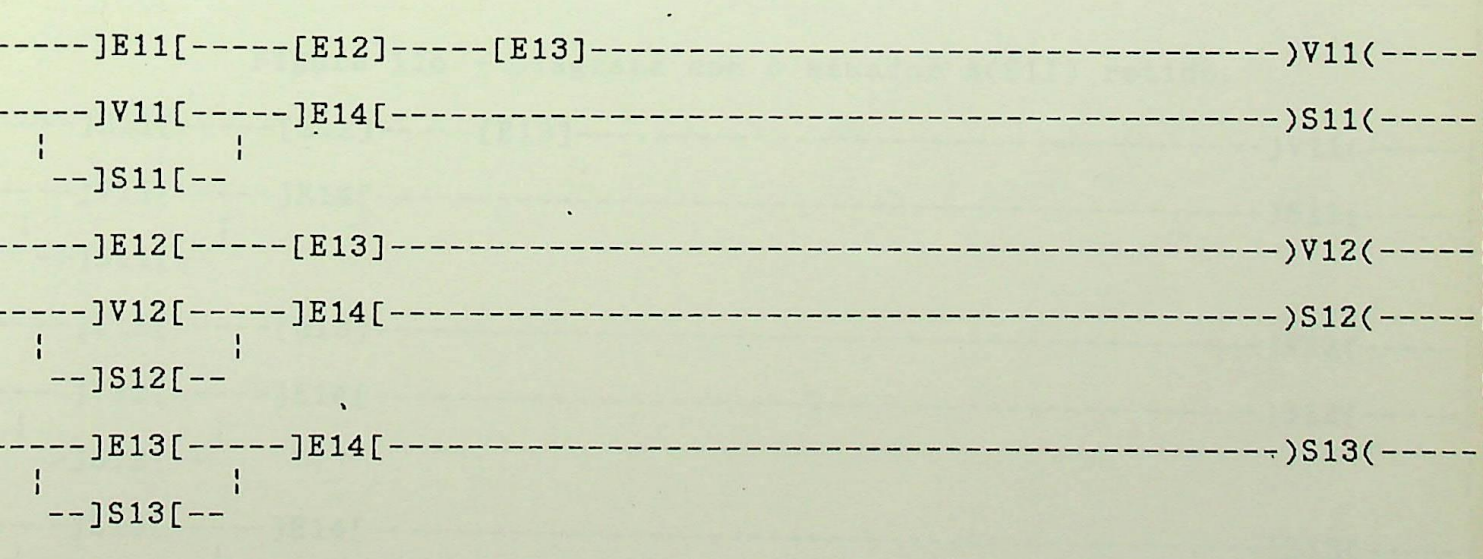


Figura 11a - Diagrama quando sensor de peça(E14) ativado.

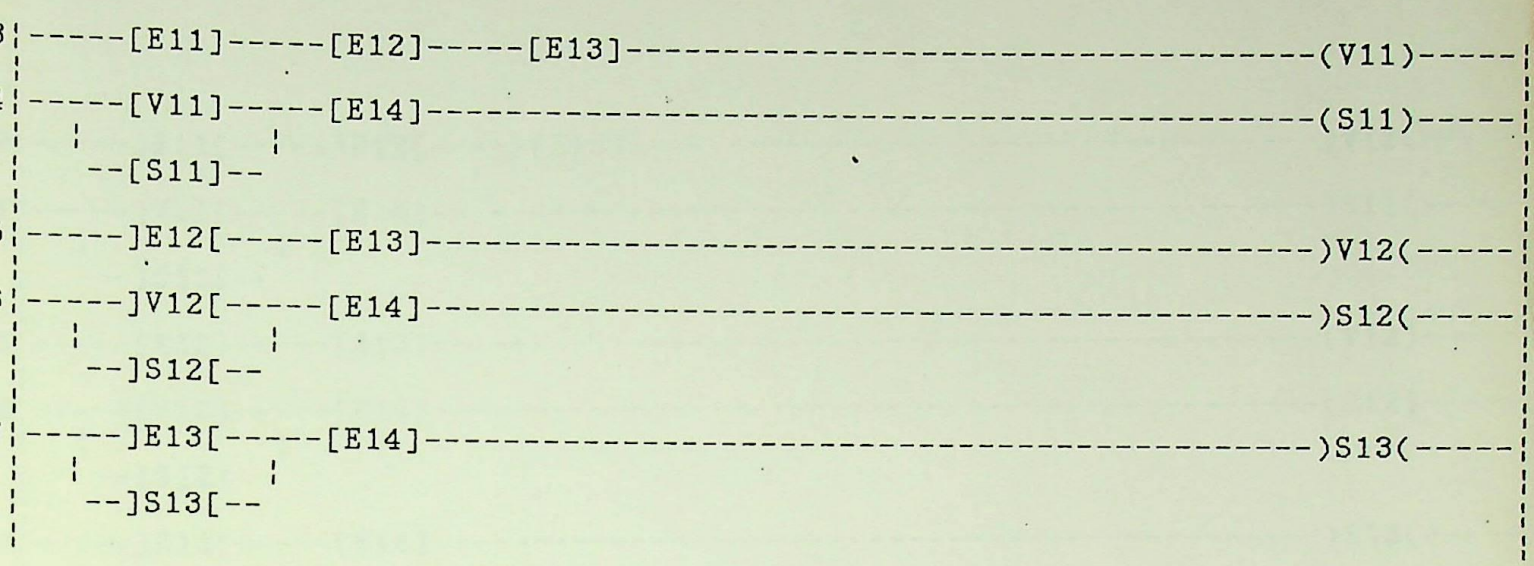


Figura 11b - Diagrama para sensor da faixa 1(E11) ativado.

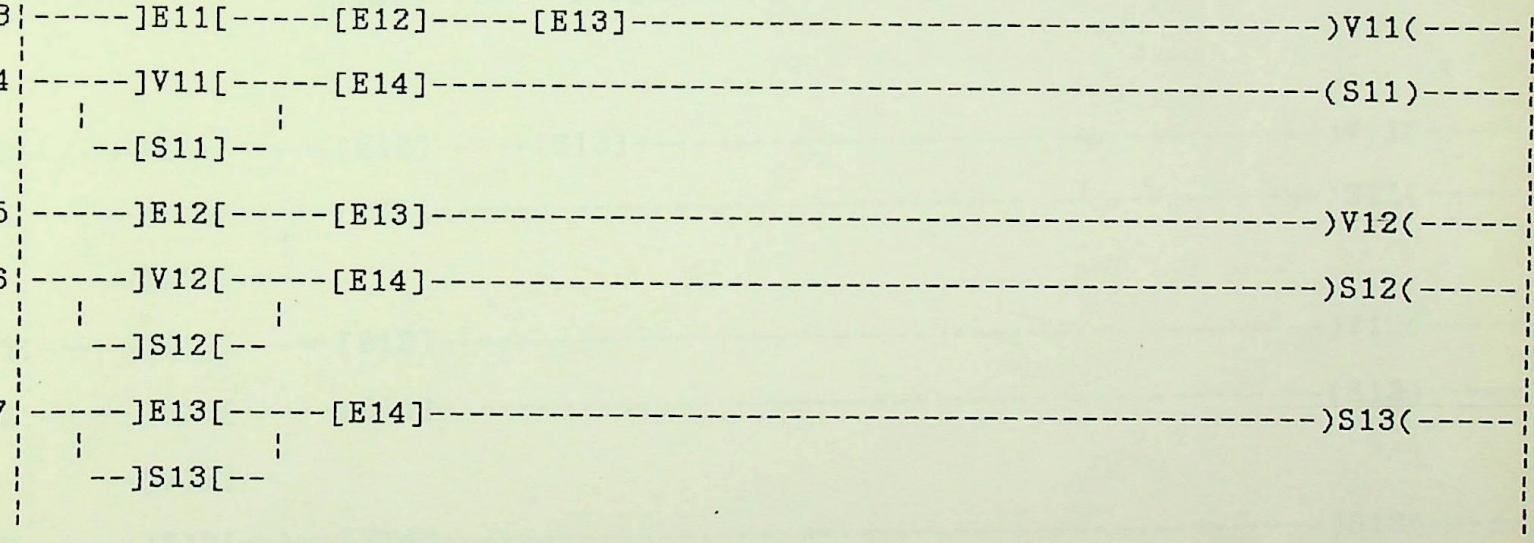


Figura 11c - Diagrama com o atuador A(S11) retido.

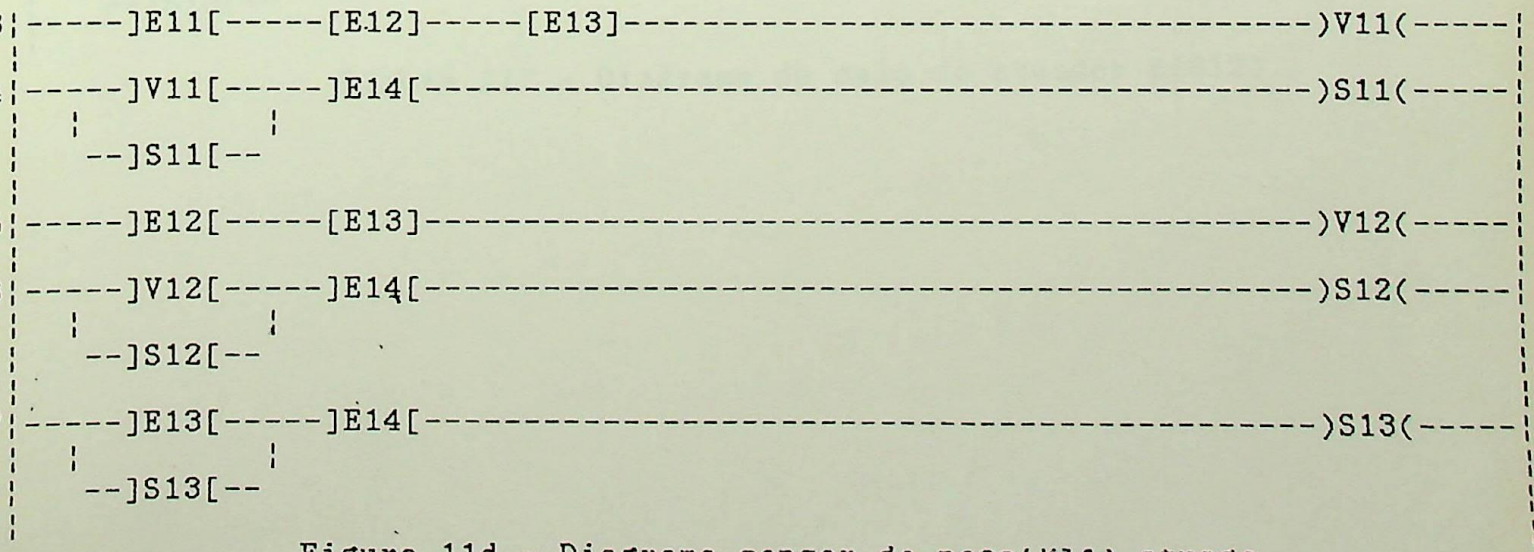


Figura 11d - Diagrama sensor de peça(E14) atuado.

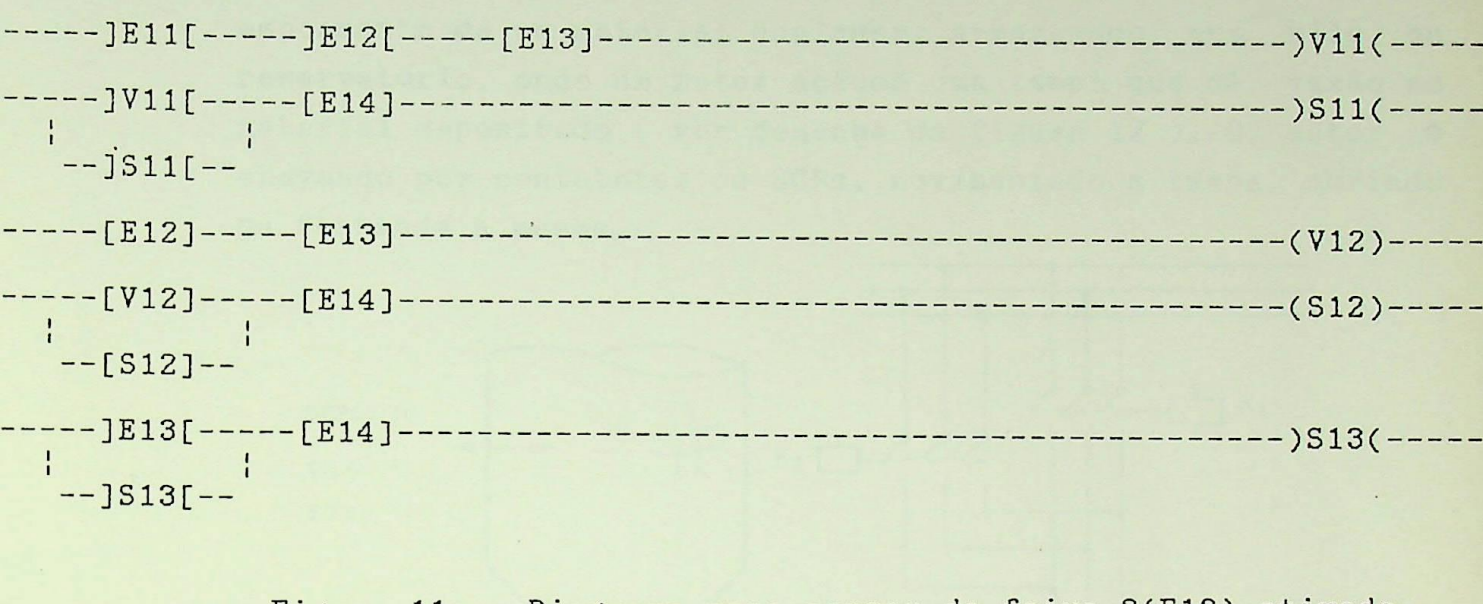


Figura 11e - Diagrama para sensor da faixa 2(E12) ativado.

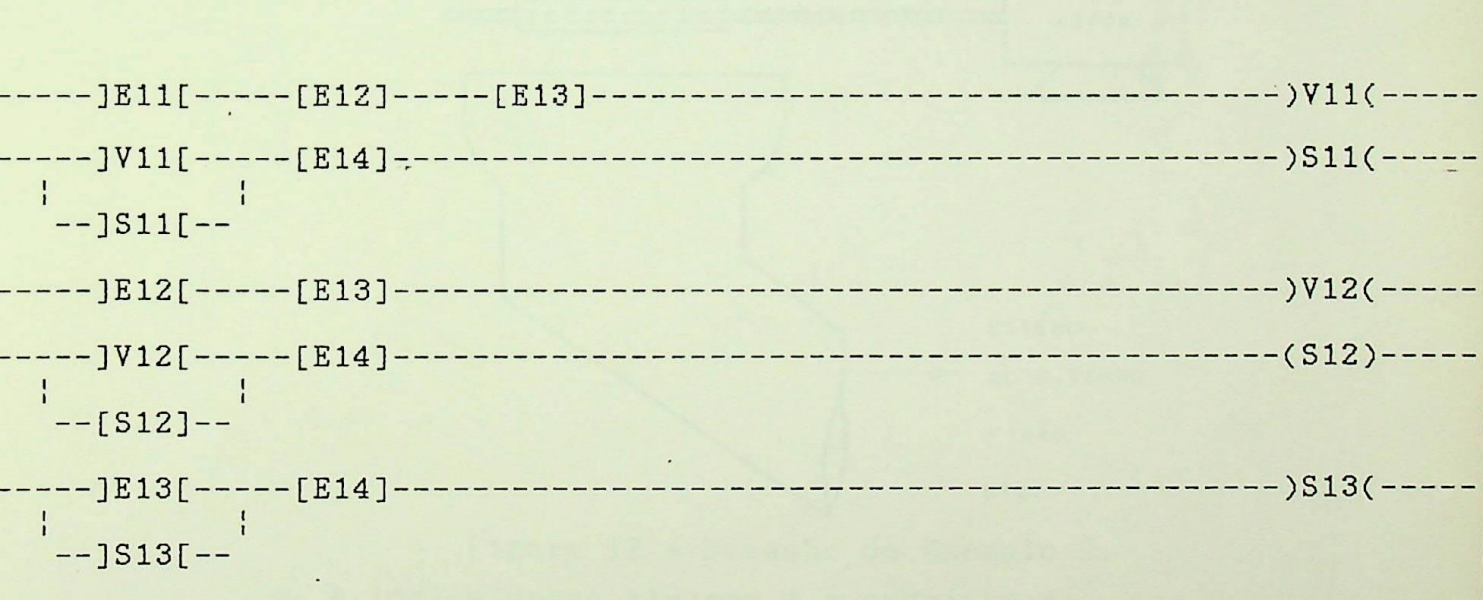


Figura 11f - Diagrama de selo do atuador B(S12).

EXEMPLO 2

Determinado processo, necessita do controle de escoamento de um material qualquer, armazenado num silo ou reservatório, onde um motor aciona uma tampa que dá vazão ao material depositado (ver desenho da figura 12). O motor é chaveado por contatores ou SCRs, movimentado a tampa, abrindo ou fechando a mesma.

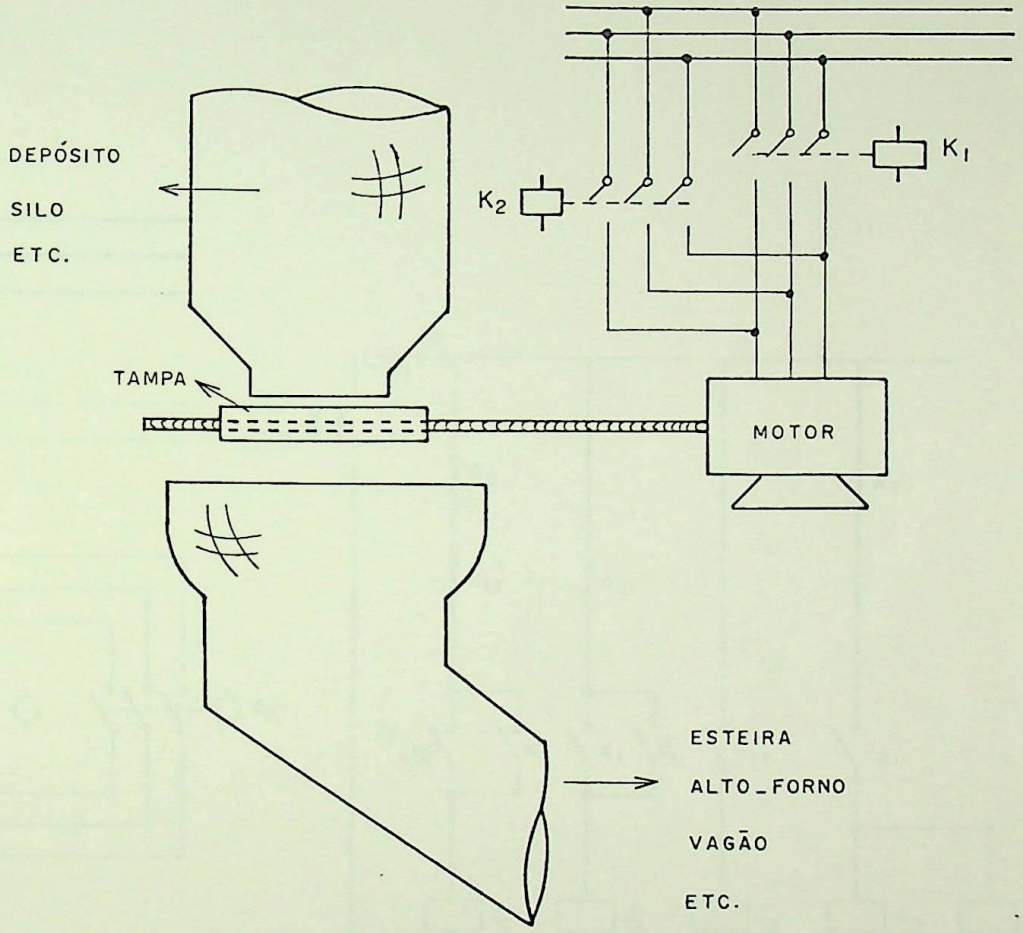


Figura 12 - Desenho do Exemplo 2.

A lógica desse sistema é a seguinte :

- A um comando inicial o motor é acionado por K1, num sentido de rotação abrindo a tampa do silo, durante um tempo t1.

- Deçorrido t1, K1 é desenergizado parando o motor, ficando a tampa aberta durante um tempo t2, determinando uma taxa de escoamento do material armazenado para o processo.

- Após t2, K2 é energizado durante um tempo t3, invertendo o sentido de rotação do motor, fechando a tampa.

- Passado t3, K2 é desenergizado parando o motor e o

sistema fica a espera de um novo comando inicial.

Na figura 13 tem-se a implementação da lógica, utilizando contatores e reles temporizados(d1, d2, d3) para os tempos t1, t2 e t3. No caso de sobrecarga o motor é desativado com a desenergização de K1 e K2 por um relé térmico(F2).

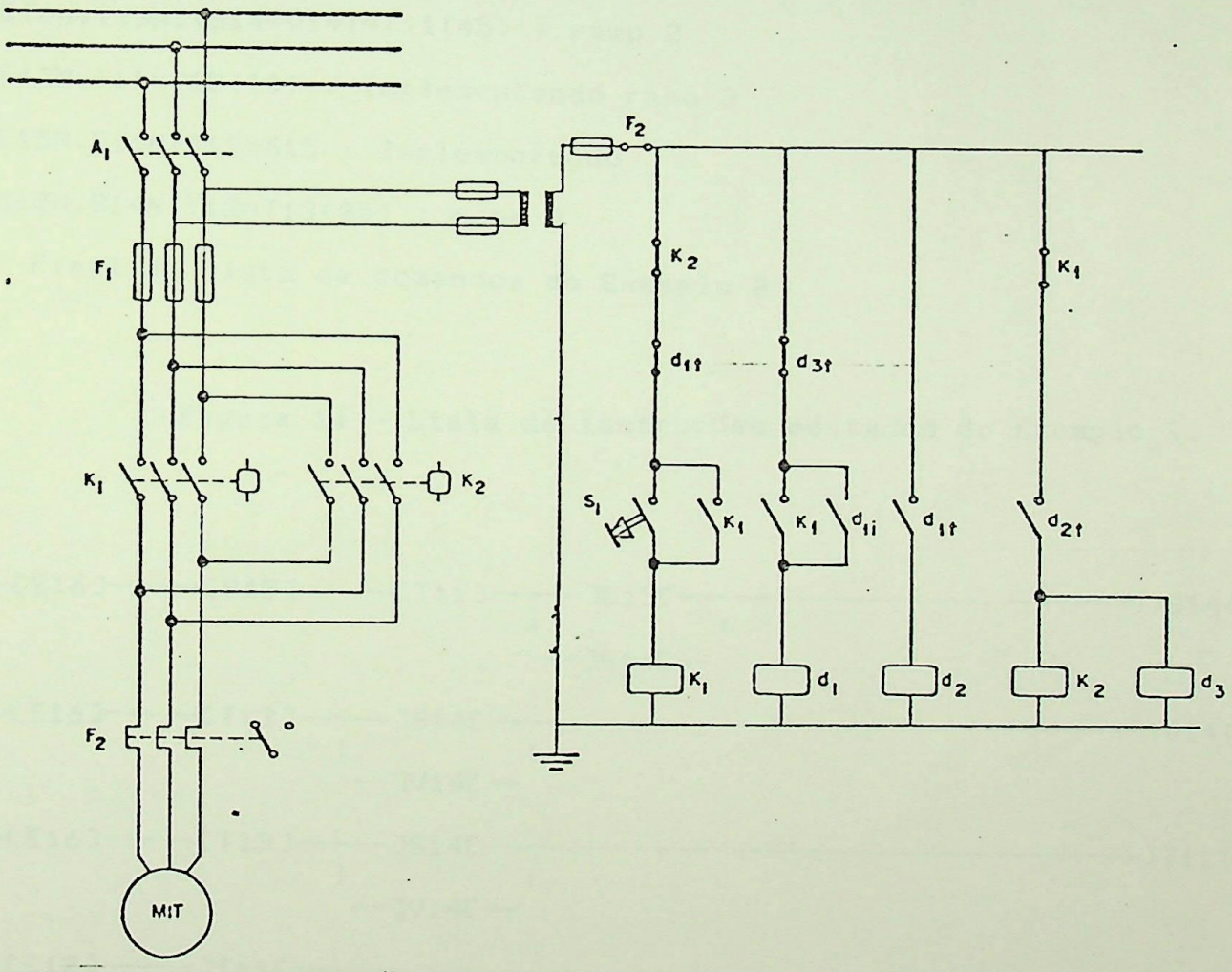


Figura 13 - Lógica de relé do Exemplo 2.

Usando o simulador, temos na figura 14 a lista de instruções dessa lógica de controle, por exemplo, com tempos de t1=5s, t2=10s e t3=5s . E nas figuras 15, os diagramas de contatos equivalentes obtidos.

```
9 ; Do Exemplo 2 : S1=E15, F2=E16, K1=S14, K2=S15, d1=T11, d2=T12, d3=T13
10 ;Lista de comandos do sequenciamento :
11 E16N.S15N.T11N.(E15+S14)=S14 ; Implementando ramo 1
12 E16N.T13N.(S14+V14)=V14 ; Implementando
13 E16N.T13N.(S14+V14)=T11(05) ; ramo 2
14 E12N.T11=T12(10) ; Implementando ramo 3
15 E15N.S14N.T12=S15 ; Implementando
16 E15N.S14N.T12=T13(05) ; ramo 4
17 ; Final da lista de comandos do Exemplo 2
18 F
```

Figura 14 - Lista de instruções editadas do Exemplo 2.

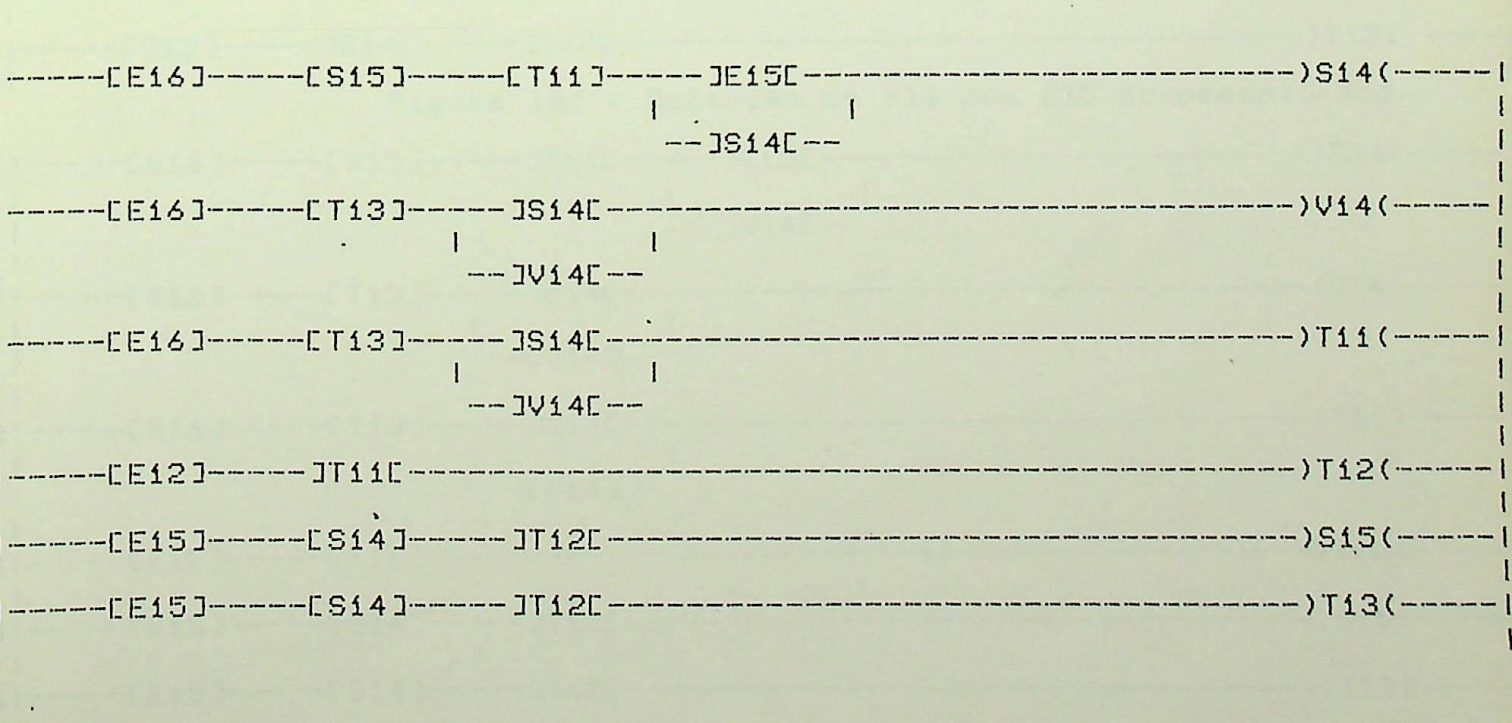


Figura 15a - Diagrama inicial.

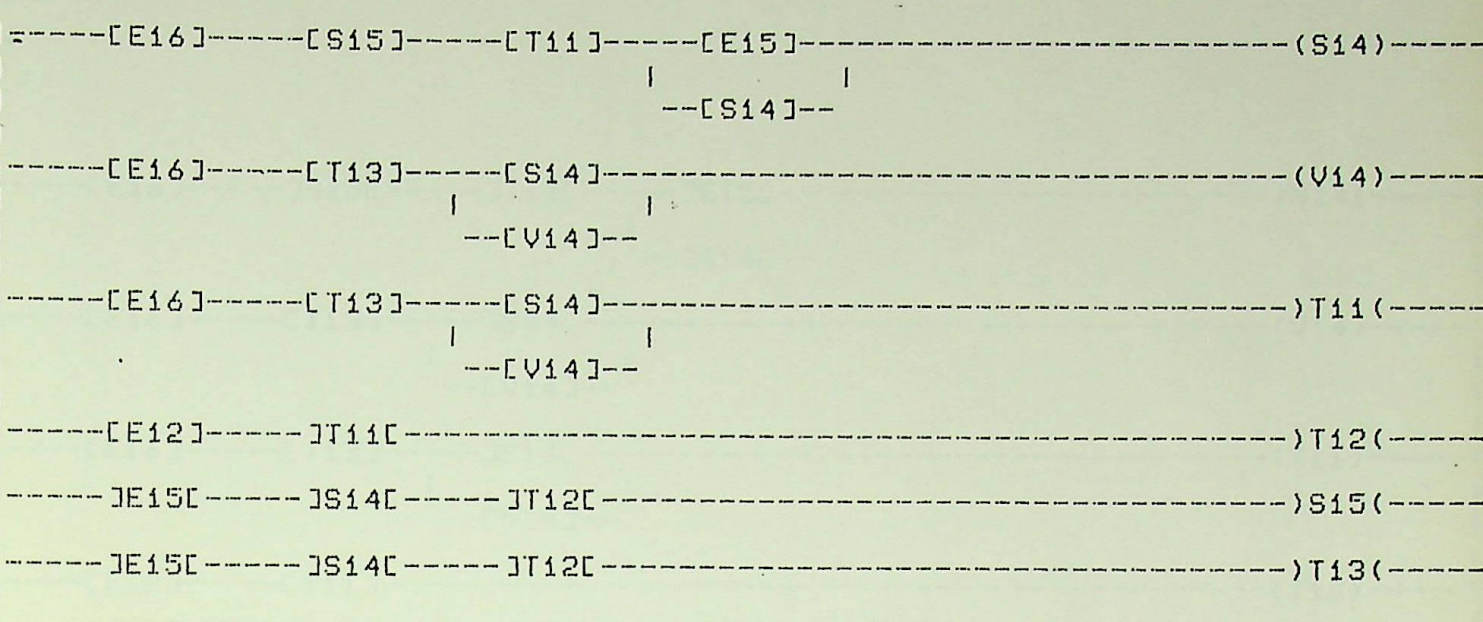


Figura 15b - Comando inicial (E15 pressionado).

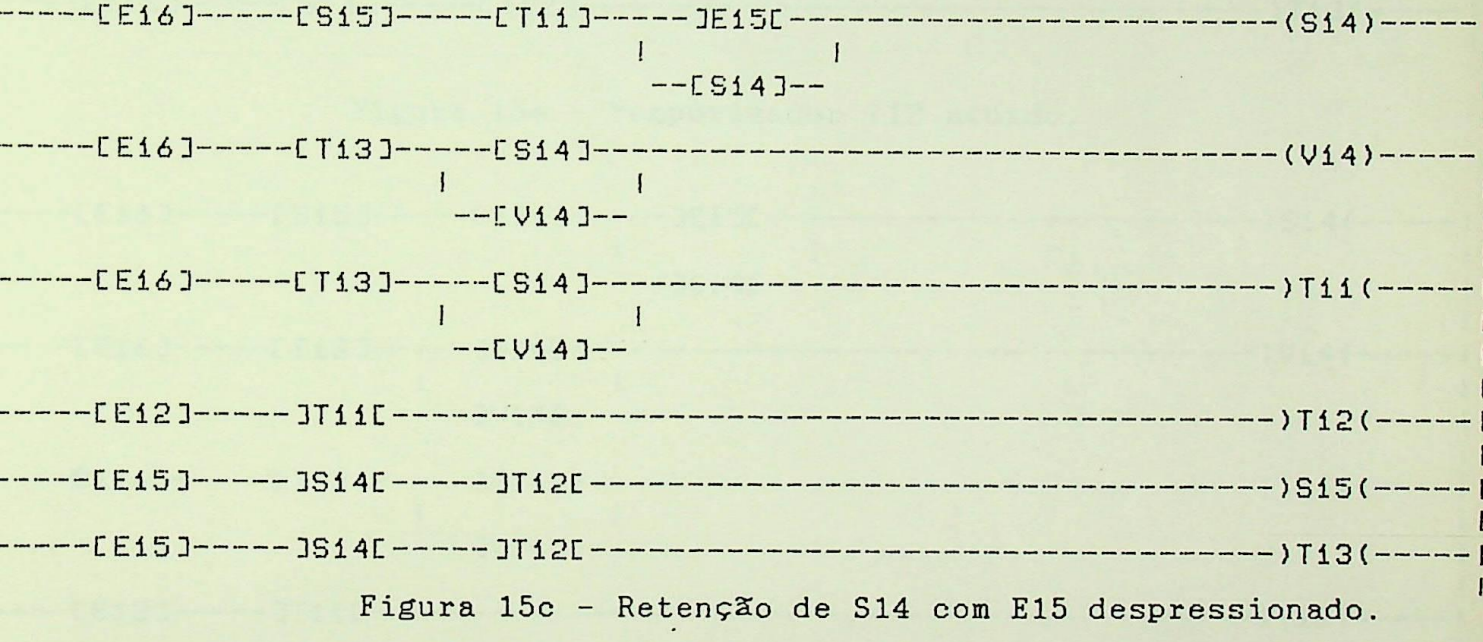


Figura 15c - Retenção de S14 com E15 despressionado.

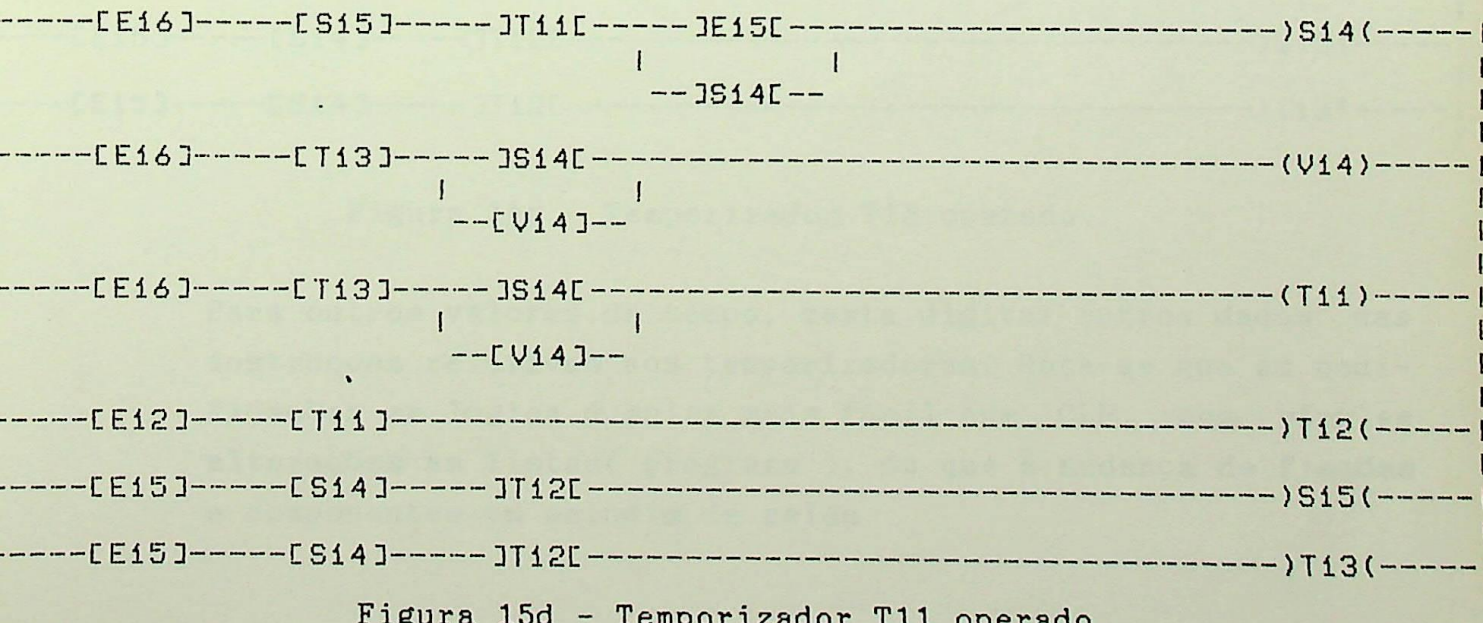


Figura 15d - Temporizador T11 operado.

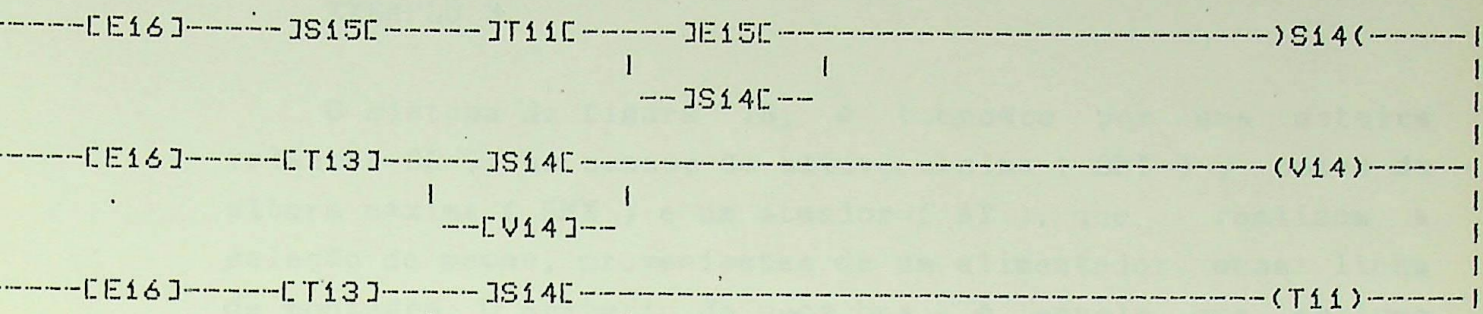


Figura 15e - Temporizador T12 atuado.

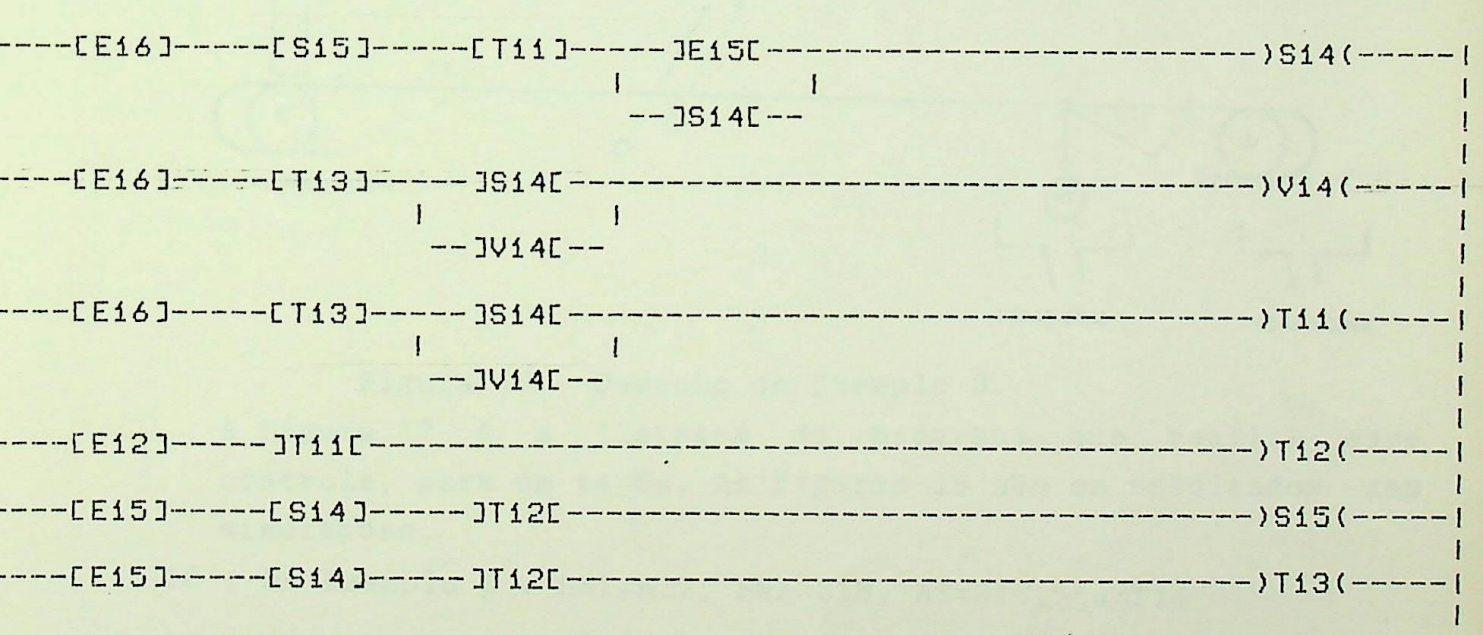


Figura 15g - Temporizador T13 operado.

Para outros valores de tempo, basta digitar outros dados nas instruçoes relativas aos temporizadores. Nota-se que as modificações em lógica é muita mais fácil num CLP, com simples alterações em listas(programa), do que a mudança de fiações e componentes em painéis de relés.

EXEMPLO 3

O sistema da figura 16, é composto por uma esteira rolante(ER), um sensor de altura mínima (SMI) , outro de altura máxima (SMX) e um atuador (AT), que realizam a seleção de peças, provenientes de um alimentador, numa linha de montagem. O critério de peça boa , é aquela que estiver entre o valor mínimo e máximo indicados pelos sensores . Toda vez que esta situação for detetada o atuador(AT) deverá ficar energizado por um tempo estabelecido.

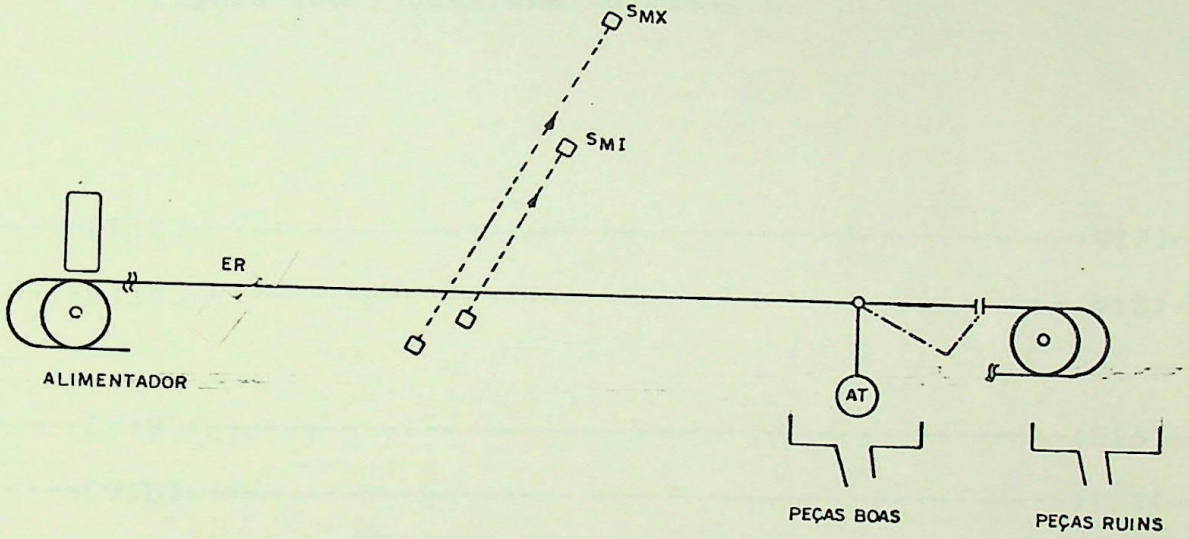


Figura 16 - Desenho do Exemplo 3.

A figura 17 é a listagem do programa que realiza esse controle, para um t4=8s. As figuras 18 são os resultados das simulações.

```
18 ; Do Exemplo 3 : SMI=E17, SMX=E18, AT=S16, t4=T14
19 ;Lista de comandos :
20 E17.E18N=V17 ; Caso de altura boa
21 (V17+V18).T14N=V18 ; Grandeza auxiliar
22 V18.V18=S16 ; Comando do atuador
23 V18.V18=T14(08) ; Temporizador
24 ; Final da lista do Exemplo 3
25 F
```

Figura 17 - Programa editado para o exemplo 3.

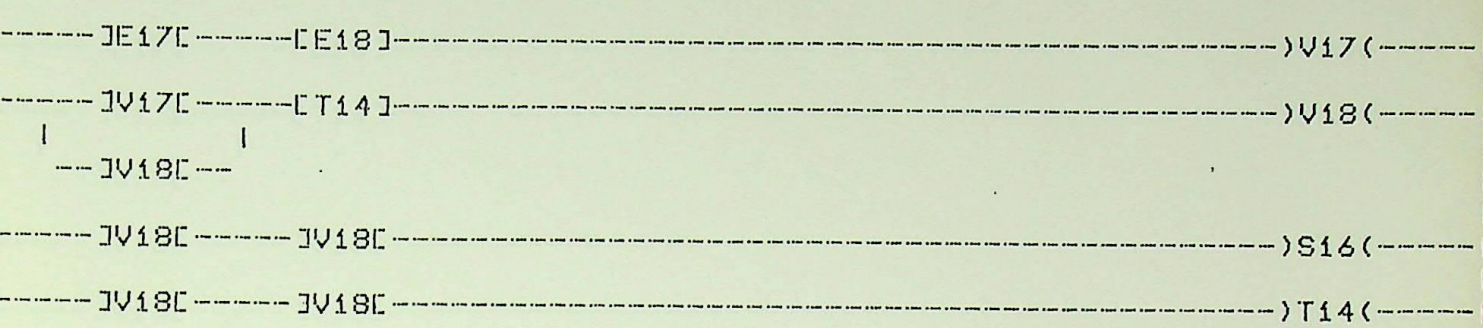


Figura 18a - Diagrama inicial.

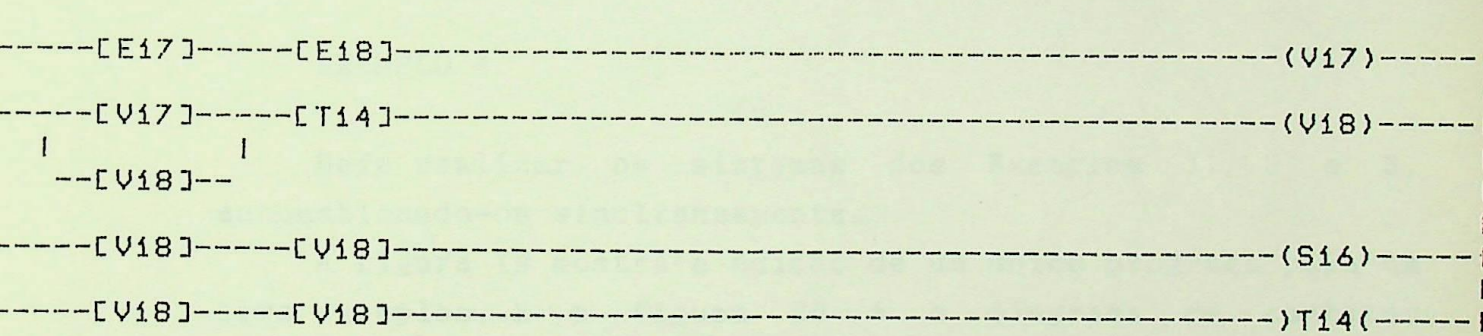


Figura 18b - Entrada(E17) de altura mínima ativada.

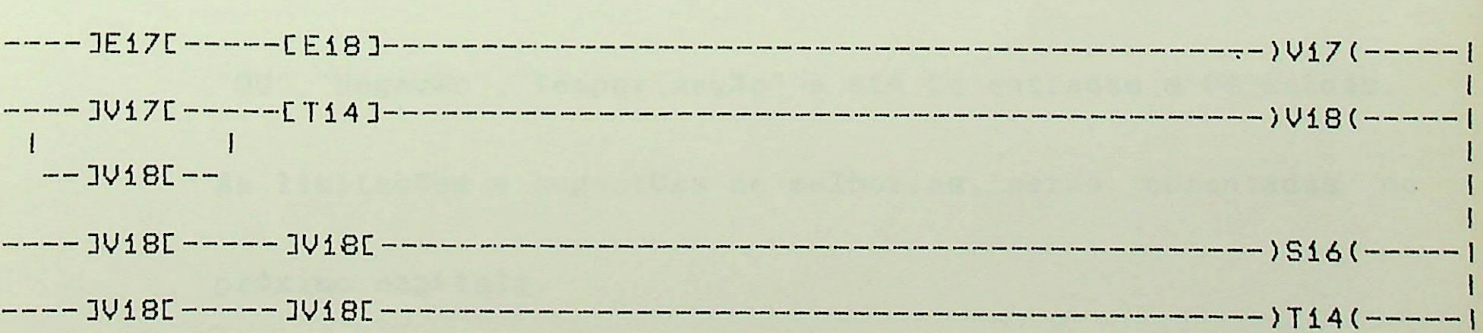


Figura 18c - Temporização da saída(S16) ativada.

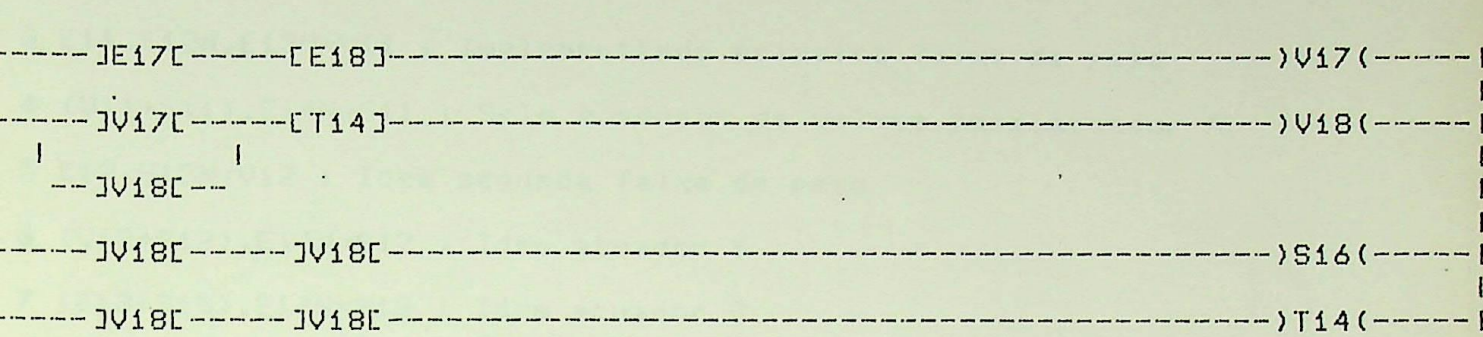


Figura 18d - Fim da temporização.

EXEMPLO 4

Seja realizar os sistemas dos Exemplos 1, 2 e 3, automatizando-os simultaneamente.

A figura 19 mostra a edição de um único programa para os três exemplos. E a figura 20 é o diagrama de contatos equivalente dos três controles seqüenciais.

Este trabalho permite a simulação de praticamente todo o controle combinacional/seqüencial , que tenha operações"E", "OU","Negação","Temporização" e até 64 entradas e 64 saídas. As limitações e sugestões de melhorias, serão comentadas no próximo capítulo.

```
1 ; Do Exemplo 1 : a=E11, b=E12, c=E13, d=E14, A=S11, B=S12, C=S13
2 ;Lista de comandos do controle :
3 E11.E12N.E13N=V11 ; Implementando primeira faixa de peso
4 (V11+S11).E14N=S11 ; Selo e sensor de volume para atuador A
5 E12.E13N=V12 ; Idem segunda faixa de peso
6 (V12+S12).E14N=S12 ; Idem atuador B
7 (E13+S13).E14N=S13 ; Idem atuador C
8 ; Final da lista de comandos do Exemplo 1
9 ; Do Exemplo 2 : S1=E15, F2=E16, K1=S14, K2=S15, d1=T11, d2=T12, d3=T13
10 ;Lista de comandos do sequenciamento :
11 E16N.S15N.T11N.(E15+S14)=S14 ; Implementando ramo 1
12 E16N.T13N.(S14+V14)=V14 ; Implementando
13 E16N.T13N.(S14+V14)=T11(05) ; ramo 2
14 E12N.T11=T12(10) ; Implementando ramo 3
15 E15N.S14N.T12=S15 ; Implementando
16 E15N.S14N.T12=T13(05) ; ramo4
17 ; Final da lista de comandos do Exemplo 2
18 ; Do Exemplo 3 : SMI=E17, SMX=E18, AT=S16, t4=T14
19 ;Lista de comandos :
20 E17.E18N=V17 ; Caso de altura boa
21 (V17+V18).T14N=V18 ; Grandeza auxiliar
22 V18.V18=S16 ; Comando do atuador
23 V18.V18=T14(08) ; Temporizador
24 ; Final da lista do Exemplo 3
25 F
```

Figura 18 - Edição dos três exemplos num só programa.

Figura 20 - Diagrama equivalente.

CAPÍTULO 6

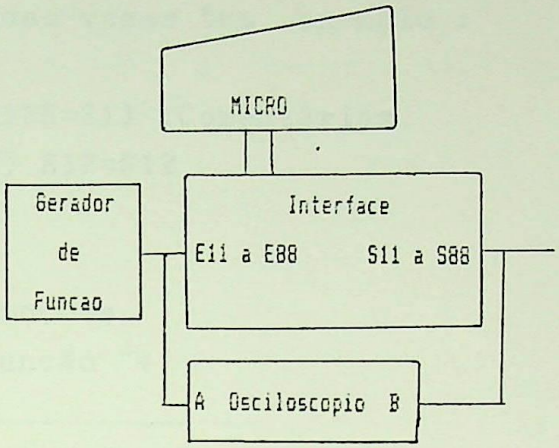
COMENTÁRIOS E CONCLUSÕES

O trabalho de dissertação apresentou algumas dificuldades, limitações e possibilidades que passaremos a relatar.

1) A maior dificuldade foi o desenvolvimento dos interpretadores de comandos : a subrotina "Lógica" que implementa as funções "E", "OU", "Negação" e "Temporização" ; e a subrotina "Sint_Relé" para geração dos diagramas de contato, no vídeo ou impressora. Esses blocos constituem a esséncia do simulador. A dificuldade é que não se encontra muita bibliografia, a respeito da construção de programas tradutores, que apresentem técnicas ou métodos fáceis de serem usados, para resolução de problemas reais. A implementação foi realizada de forma empírica, na base da tentativa e erro, entretando considerando as técnicas e recomendações de programação e desenvolvimento de software. *

2) A principal limitação do simulador de CLP, quando se utiliza o interface para controle em tempo real, é o tempo de atualização (ou varredura) das saídas digitais, de aproximadamente 5ms (cinco milisegundos) por função "E" ou "OU". Este dado foi medido, injetando-se um sinal digital através de um gerador de funções,em uma entrada do interface, realizando uma operação lógica e ativando uma saída. Utilizando um osciloscópio de dois canais, um na entrada, onde injeta-se o sinal do gerador e o outro numa saída do interface, mede-se o tempo de atualização, que é a diferença indicada nos dois canais do osciloscópio (veja esquema de medição a seguir).

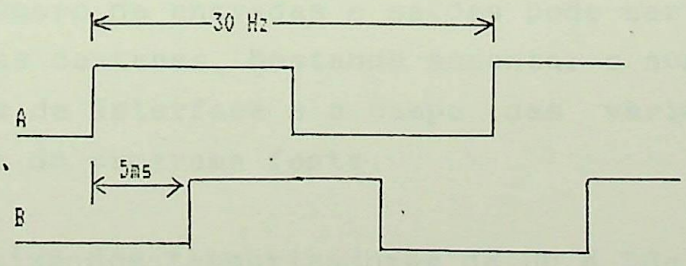
* [7]



(a) Esquema de medição.

```
1 E11.E11=S11          ou        1 (E11+E12)=S11
2 F                              2 F
```

(b) Programa de teste.



(c) Resultado.

=> Medição do tempo de varredura <=

O tempo total de um programa é o número de funções "." e "+" realizadas vezes 5ms, exemplo :

```
1 E11.E12.E13N=S11 ;Comentários.
2 (E11+E13N).E12=S12
3 F
```

Temos : 3 funções "."
1 função "+"

Total 4 funções

Portanto : Tempo de atualização = 4 x 5ms = 20ms.

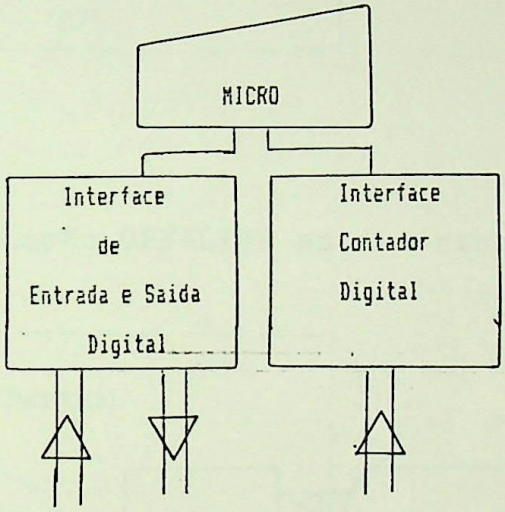
A varredura pode ser melhorada com o emprego de linguagens/compiladores mais velozes, que gerem programas objetos mais compactos, como "C", "PL/M", "Assembler", etc e sistemas operacionais de tempo real (RMX, AMX, etc.) no lugar de versões comuns.

3) O simulador pode ser incrementado em outras faixas e funções :

- O número de entradas e saídas pode ser aumentado,de 64 para algumas centenas, bastando aumentar o número de portas das placas de interface e o campo das variáveis declaradas no começo do programa fonte.

- A faixa dos temporizadores de 00 a 99s,com resolução de 1s, pode ser extendida de centésimos a milhares de segundos , uma vez que a subrotina leitura de tempo cobre essa faixa.

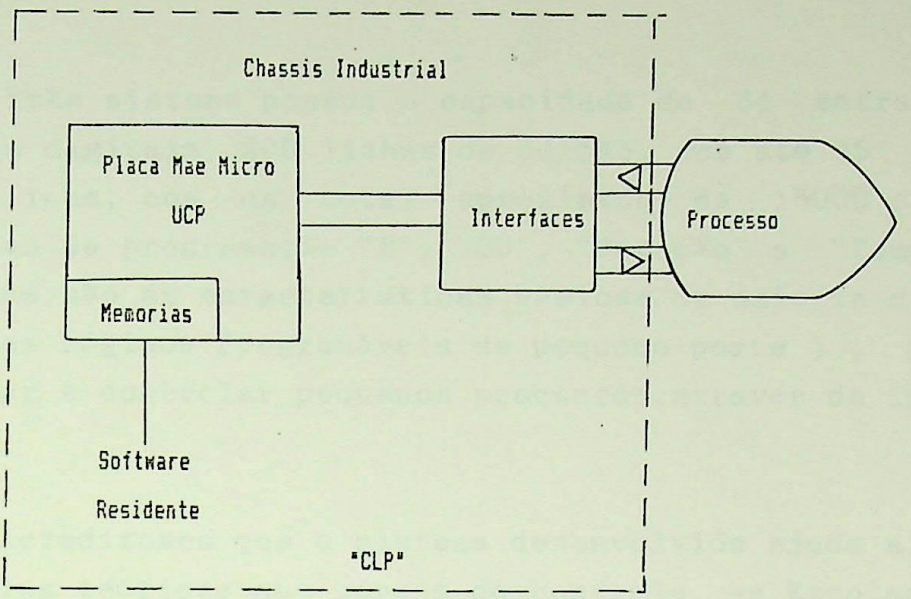
- Funções do tipo "Manipulação de bits", "Contadores" e outras podem ser implementadas, com alterações no programa fonte, mas geralmente necessitam de outros dispositivos periféricos, para não comprometer o tempo de varredura, a contagem não é realizada pela UCP (processador) do micro, mas por circuito dedicado, ver diagrama abaixo.



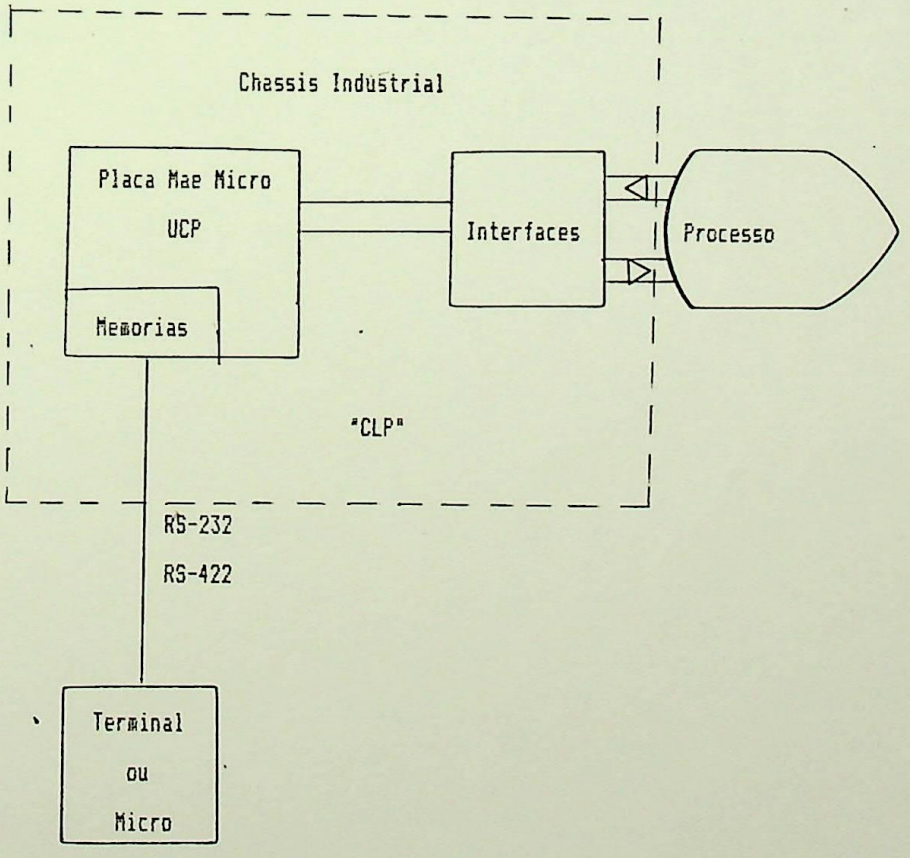
=> Inclusão de um contador. <=

- Existe a possibilidade de transformar microcomputadores em versões industriais (acondicionados em gabinetes contra ambientes agressivos), mais periféricos de entrada e saída, em controladores programáveis para uso em campo. O programa aplicativo pode ficar residente na memória do micro, ou der transferido por terminais locais ou remotos, através de interfaces de comunicação (RS-232, RS-422, etc.). A vantagem desse equipamento é que o hardware já está pronto, depurado, fabricado em escala comercial. Veja diagrama. *

* [5],[6]



(a) "CLP" com programação OFF-LINE em memórias EPROMs.



(b) "CLP" com programação ON-LINE em terminais local/remoto.

=> Micro + Interface = "CLP". <=

Este sistema possui a capacidade de 64 entradas e 64 saídas digitais, 200 linhas de edição, com até 75 caracteres por linha, com um total aproximado de 15000 caracteres, funções de programação "E", "OU", "Negação" e "Temporização" (essas são as características básicas da maioria dos Controladores Lógicos Programáveis de pequeno porte) , permitindo simular e controlar pequenos processos,através de interfaces.

Acreditamos que o sistema desenvolvido ajude a melhorar as aulas teóricas nos cursos de controle, em Escolas Técnicas e Universidade, embora o ideal é a utilização real de equipamentos industriais, seja através de promoções especiais de compra para essas instituições , ou doações das indústrias fabricantes desses equipamentos, de modo a termos profissionais recém-formados, não muito defasados com a tecnologia atual.

APÊNDICE A

MANUAL DE UTILIZAÇÃO DO SIMULADOR

Manual de Utilização do Simulador

O uso do simulador de CLP é bem simples , bastando seguir os procedimentos abaixo :

1- Instale o interface de entradas e saídas digitais, com o microcomputador desligado, mum dos slots livres do mesmo, quando se for realizar controles externos.

2- Ligue o microcomputador (compatível com a linha IBM PC-XT) com no mínimo de 256K de RAM e carregue o sistema operacional (qualquer versão igual ou superior a 2.0).

3- Coloque o disco que contenha o programa de simulação já compilado, digite SCLP e pressione <ENTER>.

4- Aparecerá no vídeo a seguinte tela (figura 21) :

```
        PROGRAMA SIMULADOR DE

   CONTROLADORES LOGICOS PROGRAMAVEIS

Projetista : Carlos Alberto Murari Pinheiro / EFEI-DON

Pressione qualquer tecla para continuar
```

Figura 21 - Tela inicial do simulador.

5- Nova tela será apresentada (figura 22) :

```
                    MENU
                    ----

   - Editor <E>
   - Simulacao no video ou na impressora <S>
   - Simulacao com Interface <I>
   - Retorno ao Menu <M>
   - Finalizar Sessao <F>
```

Figura 22 - Menu.

Pressione a tecla correspondente a opção desejada. Em listas que ainda não foram editadas, pressione <E> para Editor.

6- No caso da opção Editor <E>, aparecerá no vídeo a mensagem da figura 23 :

```
                    EDITOR
  [ Veja  as regras de sintaxe no manual de utilizacao ]

      - Limpar Buffer de Edicao, pressione <S> ou <N> :

      - Carregar programa de disco, pressione <S/N> :
```

Figura 23 - Seleção de programa em disco ou não.

Deve-se escolher <S> para limpar Buffer em programas novos, a serem editados ou quando se carrega programas gravados no disco e a opção <N> para programas já carregados,que vão ser corrigidos , modificados ou ampliados . Para se ler programas anteriormente editados e gravados, pressiona-se a opção <S>, aparecendo uma mensagem pedindo o nome do arquivo.

7- Depois da escolha "Limpar Buffer" e "Carregar programa" o sistema mostrará as opções do Editor (figura 24) :

```
   Opcoes Editor :


- Digitar Linhas <D>
- Inserir Linha <I>
- Apagar Linha <A>
- Listar <L>
- Gravar em disco <G>
- Sintaxe e diagrama de rele equivalente <R>
- Opcoes do Editor <O>
- Menu <M>
```

Figura 24 - Opções do Editor.

8- Na escolha de Digitar linhas <D> para programas novos, uma mensagem pedira o número da linha a se iniciar a digitação, a partir daí basta digitar a lista de instruções correspondente ao controle e comentários , se houver (veja figura 25) :

```
)igite o N da Linha ( 1 a 200 ) e pressione <ENTER> :1

)igite as Instrucoes ( ate a marca limitadora = ! )
)or linha e pressione <ENTER>

'ara deixar Insercao digite espaco em branco
 <ENTER> na linha posterior a ultima linha digitada

 E11.E12=S11

 E11N.E13=S12 ; Comentarios
```

Figura 25 - Procedimentos de digitação.

As escolhas Apagar Linha <A>, Inserir Linha <I>, Listar <L>, Gravar em disco <G>, Opcoes do Editor <O> são imediatas. Pressionando <M> duas vezes volta-se ao Menu.

9- A opção Sintaxe e diagrama de relé equivalente <R> , faz a verificação de sintaxe da linha em questão e gera no vídeo ou impressora o diagrama de contatos correspondente a lista de instruções, figura 26a :

```
Pressione <I> para Impressora se nao outra tecla
Ligue a Impressora e depois pressione uma tecla

igite o N da Linha ( 1 a 200 ) e pressione <ENTER> :1

 E11.E12=S11

|-----]E11[-----]E12[-----------------------------------------)S11(-----
|
```

Figura 26a - Diagrama de relé equivalente.

10- <S> para Simulação no vídeo/impressora gera as sequintes mensagens e diagramas (figura 26b) :

```
ressione <E>=Status das Entradas e <P>=Parar :

gite separando com espaco em branco,e depois <ENTER> o :
Conector[1 a 8]; N Borne[1 a 8] e o Estado[0 ou 1]

1 1 1
1 2 1
1 3 0

-----[E11]-----[E12]-----------------------------------------(S11)-----

-----]E11[-----]E13[-----------------------------------------)S12(-----

ressione <P>=Parar, outra <Tecla>=Continuar Simulacao
```

Figura 26b - Simulação.

A simulação do estado das entradas digitais é feito pelo teclado, conforme indicado.

11- <I> para Simulação com Interface, testa a sintaxe do programa de aplicação, o executa lendo as entradas digitais e atualiza as saídas digitais. Nesse modo o interface deve estar instalado. Em operação é mostrada as mensagens a seguir:

```
..... Teste Sintaxe Programa ......

..... Programa sem Erros de Sintaxe ......

:) Pressione uma tecla para comecar execucao
 Para bloquear a execucao pressione a tecla (B)

..... Execucao Iniciada ......
```

12- Para se deixar o simulador e retornar ao sistema operacional pressionar a tecla <F>, finalizando a sessão.

APÊNDICE B

LISTAGEM DO PROGRAMA SIMULADOR

```
rogram SCLP;

{ Este programa e a implementacao de um simulador }

{ de Controladores Logicos Programaveis }

{ Versao 1.0 }

{ Programador : Carlos Alberto Murari Pinheiro }

{ Novembro 1988 }


          { Declaracao de variaveis globais :- }

          { Vetor = Matriz de 8 X 8 para armazenar estados de Entradas , }
          { Variaveis auxiliares, Temporizadores e Saidas.               }
          { Lin = Linha de instrucao com ate 75 caracteres.              }
          { Texto = Matriz de Linhas de instrucoes com ate 200 elementos.}
          { VetorS = Matriz de 8 X 8 para armazenar os valores de tempo. }
          { tecla e tec = Variveis para leitura do teclado.              }
          { Err = Variavel para indicacao de erro.                       }
          { Op_I = Variavel de escolha da opcao para impressora.         }
          { E = Entrada em lista de instrucoes nos programas editados.   }
          { V = Variavel auxiliar nas listas de instrucoes.              }
          { T = Temporizadores.                                          }
          { S = Saida de atuacao nos programas aplicativos.              }
          { EST = Estado das entradas(E),saidas(S),variaveis(V),Tempo(T).}
          { LI = Indice para linha inicial ou atual.                     }
          { Si = Valor de tempo.                                         }

Type Vetor  = Array [1..8,1..8] Of Byte;   { Declaracao de tipos }
       Lin  = String [75];
      Text  = Array [1..200] Of String [75];
     VetorS = Array [1..8,1..8] Of Real;

Var tecla,tec,Err,Op_I : char;             { Variaveis do simulador e }
              E,V,T,S : Vetor;             { identificacao de tipos.  }
                Linha : Lin;
                Texto : Text;
               EST,LI : Byte;
                   Si : Real;
                    D : VetorS;



Procedure Teclado ( Var ttec : Char ); { Subrotina de leitura de teclado }

  Begin

    Read(Kbd,ttec);      { Leitura de teclado }
    ttec:=UpCase(ttec);  { Caracter Maiusculo }

  End;    { Final subrotina de leitura de teclado }
```

```
Procedure Num_Linha ( Var LLI : Byte );  { Subrotina Numero da Linha }

  Begin

    WriteLn;WriteLn;
    Repeat
      Write('  Digite o Nº da Linha ( 1 a 200 ) e pressione <ENTER> :');
      Read ( LLI );
      WriteLn;
    Until ( LLI >= 1 ) And ( LLI <= 200 );

  End;    { Final Subrotina Numero da Linha }




Procedure Tela_Op_Ed;  { Tela de opcoes do editor }


  Begin

    ClrScr;                  { Limpar tela }
    GoToXY(28,4);            { Posicionar cursor }
    Write('Opcoes Editor :'); { Imprimir opcoes }
    GoToXY(26,8);
    Write('- Digitar Linhas <D>');
    GoTOXY(26,10);
    Write('- Inserir Linha <I>');
    GoToXY(26,12);
    Write('- Apagar Linha <A>');
    GoToXY(26,14);
    Write('- Listar <L>');
    GoToXY(26,16);
    Write('- Gravar em disco <G>');
    GoToXY(26,18);
    Write('- Sintaxe e diagrama de rele equivalente <R>');
    GoToXY(26,20);
    Write('- Opcoes do Editor <O>');
    GoToXY(26,22);
    WriteLn('- Menu <M>');

  End;  { Fim do Bloco Opcoes Editor }
```

```
Procedure Status ( Var EX : Vetor );  { Bloco para leitura de Entradas(E), }
                                      { pelo teclado.                     }

 { Variaveis locais : }

 { NCO = Numero do conector, placa, endereco, etc, ( 1 a 8 ).             }
 { NES = Numero da entrada, saida, variavel e temporizador ( 1 a 8 ).     }
 { ESTY = EST.                                                            }

 Var    NCO,NES,ESTY : Byte;

 Begin

   WriteLn('  Pressione <E>=Status das Entradas e <P>=Parar :');
   WriteLn;
   WriteLn('  Digite separando com espaco em branco,e depois <ENTER> o :');
   WriteLn('  Nº Conector[1 a 8]; Nº Borne[1 a 8] e o Estado[0 ou 1]');
   WriteLn;
   Teclado ( tec );
   While ( tec <> 'P' ) Do
     begin
       Write(tec,' : ');
       ReadLn(NCO,NES,ESTY);  { Leitura pelo teclado }
       If ( tec = 'E' ) And ( NCO >= 1 ) And ( NCO <= 8 )
                        And ( NES >= 1 ) And ( NES <= 8 )
                        And ( ESTY >= 0 ) And ( ESTY <= 1 ) Then
         EX[NCO,NES]:=ESTY;   { Atribuicao a entrada simulada }
       Teclado ( tec );
     end;
   WriteLn;WriteLn;

 End;     { Fim do Bloco leitura das entradas(E) pelo teclado }
```

```
{Bloco para determinar estado das entradas,variaveis,temporizadores,saidas}

Procedure OnOrOff (     HX : Byte;
                    Var ESTX : Byte;
                        EX,VX,TX,SX : Vetor;
                        LinhaX : Lin        );

 Var NCOX,NESX : Byte;

 Begin

   If ( LinhaX[HX+1] = '1' ) Then NCOX:=1;  { Caracteres para numeros }
   If ( LinhaX[HX+1] = '2' ) Then NCOX:=2;
   If ( LinhaX[HX+1] = '3' ) Then NCOX:=3;
   If ( LinhaX[HX+1] = '4' ) Then NCOX:=4;
   If ( LinhaX[HX+1] = '5' ) Then NCOX:=5;
   If ( LinhaX[HX+1] = '6' ) Then NCOX:=6;
   If ( LinhaX[HX+1] = '7' ) Then NCOX:=7;
   If ( LinhaX[HX+1] = '8' ) Then NCOX:=8;
   If ( LinhaX[HX+2] = '1' ) Then NESX:=1;
   If ( LinhaX[HX+2] = '2' ) Then NESX:=2;
   If ( LinhaX[HX+2] = '3' ) Then NESX:=3;
   If ( LinhaX[HX+2] = '4' ) Then NESX:=4;
   If ( LinhaX[HX+2] = '5' ) Then NESX:=5;
   If ( LinhaX[HX+2] = '6' ) Then NESX:=6;
   If ( LinhaX[HX+2] = '7' ) Then NESX:=7;
   If ( LinhaX[HX+2] = '8' ) Then NESX:=8;
   If ( LinhaX[HX] = 'E' ) Then ESTX:=EX[NCOX,NESX];
   If ( LinhaX[HX] = 'V' ) Then ESTX:=VX[NCOX,NESX];
   If ( LinhaX[HX] = 'T' ) Then ESTX:=TX[NCOX,NESX];
   If ( LinhaX[HX] = 'S' ) Then ESTX:=SX[NCOX,NESX];

 End;  { Fim subrotina Status }
```

```
{ Subrotina de montagem de diagrama de contatos, a partir de lista de }
{ instrucoes e verificacao de sintaxe.                                 }

Procedure Sint_Rele ( LI : Byte;
                      Linha : Lin;
                      Var Err : Char;
                      Op_I : Char );

{ Declaracao de variaveis locais : }

{ H = Indice de uso geral.                                             }
{ I = Apontador de caracteres para linha de instrucoes.                }
{ J = Apontador para coluna do video.                                  }
{ K = Idem H.                                                          }
{ N = Idem I.                                                          }
{ M = Idem J.                                                          }
{ F = Indicador de contato paralelo( funcao "OU" ) nos diagramas.      }
{ C = Indicador de erro.                                               }
{ L = Apontador de caracter do contato paralelo equivalente.           }
{ Ti = Apontador de coluna do video para comeco de contato paralelo.   }
{ Tf = Apontador de coluna do video para final de contato paralelo.    }

Type  Num = '0'..'9';
      Cod = Set Of Num;

Var
  H,I,J,K,N,M : Byte;
        F,C : Char;
     L,Ti,Tf : Array [1..7] Of Byte;
   Temp,Codg : Cod;

Begin

Err:='N';
Temp:=['0'..'9'];        { Faixa de algarismos do temporizador.        }
Codg:=Temp - ['0','9']; { Faixa de algarismos de E,V,S,T.             }
I:=1;                    { Primeiro caracter de uma linha de instrucao. }
If ( Linha[I] <> ' ' ) And ( Linha[I] <> ';' ) And
   ( Linha[I] <> 'F' ) Then    { Teste de espaco em branco, finalizador }
   Begin                       { de comentarios e final de linha.       }
     For N:=1 To 7 Do
       Begin
         L[N]:=0;
         Ti[N]:=0;
         Tf[N]:=0;
       end;
     N:=1;
     If ( LI <  10  ) Then
       Begin
         Write('  ',LI,'¦');
         If ( Op_I = 'S' ) Then Write(Lst,'  ',LI,'¦');
        end;
     If ( LI >= 10  ) And ( LI <=  99 ) Then
       Begin
         Write(' ',LI,'¦');
         If ( Op_I = 'S' ) Then Write(Lst,' ',LI,'¦');
       end;
```

```
If ( LI >= 100 ) And ( LI <= 200 ) Then
  begin
    Write(    LI,'|');
    If ( Op_I = 'S' ) Then Write(Lst,LI,'|');
  end;
C:=' ';
J:=4;
Repeat
  { Teste de caracteres de uma linha de instrucoes. }
  If ( Linha[I] = '.' ) Then I:=I+1; { Incrementar apontador de }
  If ( Linha[I] = '(' ) Then I:=I+1; { caracter de linha.       }
  If ( Linha[I] <> 'E' ) And ( Linha[I] <> 'V' ) And
     ( Linha[I] <> 'S' ) And ( Linha[I] <> 'T' ) Then C:='E'
   Else
     begin
       { Imprimir diagrama de contato. }
       Write('-----');
        If ( Op_I = 'S' ) Then Write(Lst,'-----');
       J:=J+5;  { Incrementar Apontador de coluna do video. }
       H:=I;
       OnOrOff ( H,EST,E,V,T,S,Linha ); { Subrotina de Estado }
       If Linha[I+3] = 'N' Then  { Teste de "Negacao" }
         begin
           If EST = 0 Then EST:=1
             Else EST:=0;
         end;
       If ( EST = 0 ) Then
         begin
           Write(']');
           If ( Op_I = 'S' ) Then Write(Lst,']');
           J:=J+1;
         end
         Else
           begin
             Write('[');
             If ( Op_I = 'S' ) Then Write(Lst,'[');
             J:=J+1;
           end;
      If ( Linha[I+1] In Codg ) Then { Teste faixa de algarismos }
        Else C:='E';
      If ( Linha[I+2] In Codg ) Then
        Else C:='E';
      For K:=1 To 3 Do
        begin
          Write(Linha[I]);
          If ( Op_I ='S' ) Then Write(Lst,Linha[I]);
          J:=J+1;
          I:=I+1;
        end;
      If ( EST = 0 ) Then
        begin
          Write('[');
          If ( Op_I = 'S' ) Then Write(Lst,'[');
          J:=J+1;
        end
        Else
          begin
            Write(']');
            If ( Op_I = 'S' ) Then Write(Lst,']');
            J:=J+1;
          end;
```

```
        If ( Linha[I] = 'N' ) Then I:=I+1;
        If ( Linha[I] = '.' ) Then I:=I+1;
        If ( Linha[I] = '+' ) Then
          begin
            L[N]:=I+1;
            Ti[N]:=J-7;
            Tf[N]:=J+3;
            If ( Linha[I+1] = 'E' ) Or ( Linha[I+1] = 'S' ) Or
               ( Linha[I+1] = 'V' ) Or ( Linha[I+1] = 'T' ) Then
              begin
                If ( Linha[I+2] In Codg ) Then
                  Else C:='E';
                If ( Linha[I+3] In Codg ) Then
                  Else C:='E';
              end
              Else C:='E';
            I:=I+4;
            If ( Linha[I] = 'N' ) Then I:=I+1;
            If ( Linha[I] = ')' ) Then I:=I+1
              Else C:='E';
          end;
        N:=N+1;
      end;
Until ( Linha[I] = '=' ) Or ( J >= 65 ) Or ( I >= 75 ) Or
      ( C = 'E' );
{ Teste final expressao ou condicao maxima dos apontadores J , I e }
{ indicacao de erro.                                               }

If ( Linha[I] <> '=' ) Or ( J >= 65 ) Or ( I >= 75 ) Then C:='E'
  Else
    begin
      I:=I+1;
      While ( J < 64 ) Do
        begin
          Write('-');
          If ( Op_I = 'S' ) Then Write(Lst,'-');
          J:=J+1;
        end;
      Write('-----');
       If ( Op_I = 'S' ) Then Write(Lst,'-----');
        H:=I;
        OnOrOff ( H,EST,E,V,T,S,Linha );
        If ( EST = 0 ) Then
          begin
            Write(')');
            If ( Op_I = 'S' ) Then Write(Lst,')');
          end
          Else
            begin
              Write('(');
              If ( Op_I = 'S' ) Then Write(Lst,'(');
            end;
      If ( Linha[I] = 'S' ) Or ( Linha[I] = 'V' ) Or
         ( Linha[I] = 'T' ) Then
```

```
        begin
          If ( Linha[I+1] In Codg ) Then
            Else C:='E';
          If ( Linha[I+2] In Codg ) Then
            Else C:='E';
        end
        Else C:='E';
      If ( Linha[I] = 'T' ) Then
        begin
          If ( Linha[I+4] In Temp ) Then  { Teste faixa algarismos }
            Else C:='E';
          If ( Linha[I+5] In Temp ) Then
            Else C:='E';
        end;
      For K:=1 To 3 Do
        begin
          Write(Linha[I]);
          If ( Op_I = 'S' ) Then Write(Lst,Linha[I]);
          I:=I+1;
        end;
        If ( EST = 0 ) Then
          begin
            Write('(');
            If ( Op_I = 'S' ) Then Write(Lst,'(');
          end
          Else
            begin
              Write(')');
              If ( Op_I = 'S' ) Then Write(Lst,')');
            end;
      Write('-----');
      Write('¦');
      If ( Op_I ='S' ) Then
        begin
          Write(Lst,'-----');
          WriteLn(Lst,'¦');
        end;
    end;

{ Rotina para diagramar contatos paralelos }
F:='N';
For K:=1 To 6 Do
  If ( Ti[K] <> 0 ) Then F:='S';
If ( F = 'S' ) Then
begin
Write('    ¦');
If ( Op_I = 'S' ) Then Write(Lst,'    ¦');
M:=4;
N:=1;
Repeat
  If ( Ti[N] <> 0 ) Then
    begin
      While ( M < Ti[N]-1 ) Do
        begin
          Write(' ');
          If ( Op_I = 'S' ) Then Write(Lst,' ');
          M:=M+1;
        end;
      Write('¦');
      If ( Op_I = 'S' ) Then Write(Lst,'¦');
      M:=M+1;
    end;
```

```
    If ( Tf[N] <> 0 ) Then
      begin
        Repeat
          Write(' ');
          If ( Op_I = 'S' ) Then Write(Lst,' ');
          M:=M+1;
        Until ( M >= Tf[N]-1 );
        If ( Tf[N] <> Ti[N+1] ) Then
          begin
            Write('¦');
            If ( Op_I = 'S' ) Then Write(Lst,'¦');
            M:=M+1;
          end;
      end;
    N:=N+1;
  Until ( N = 7 );
  If ( L[7] <> 0 ) Then
    begin
      If ( Tf[6] = 0 ) Then
        begin
          Repeat
            Write(' ');
            If ( Op_I = 'S' ) Then Write(Lst,' ');
            M:=M+1;
          Until ( M >= 66 );
          Write('¦');
          If ( Op_I = 'S' ) Then Write(Lst,'¦');
          M:=M+1;
        end;
      Repeat
        Write(' ');
        If ( Op_I = 'S' ) Then Write(Lst,' ');
        M:=M+1;
      Until ( M >= 76 );
      Write('¦');
      If ( Op_I = 'S' ) Then Write(Lst,'¦');
      M:=M+1;
    end;
  Repeat
    Write(' ');
    If ( Op_I = 'S' ) Then Write(Lst,' ');
    M:=M+1;
  Until ( M >= 79 );
  Write('¦');
  Write('   ¦');
  If ( Op_I = 'S' ) Then
    begin
      WriteLn(Lst,'¦');
      Write(Lst,'   ¦');
    end;
  M:=4;
  N:=1;
  Repeat
    If ( Ti[N] <> 0 ) Then
      begin
        While ( M < Ti[N]-1 ) Do
          begin
            Write(' ');
            If ( Op_I = 'S' ) Then Write(Lst,' ');
            M:=M+1;
          end;
```

```
        Write(' ');
        If ( Op_I = 'S' ) Then Write(Lst,' ');
        M:=M+1;
        Write('--');
        If ( Op_I = 'S' ) Then Write(Lst,'--');
        M:=M+2;
        H:=L[N];
        OnOrOff ( H,EST,E,V,T,S,Linha );
        If Linha[H+3] = 'N' Then
          begin
            If EST = 0 Then EST:=1
              Else EST:=0;
          end;
        If ( EST = 0 ) Then
          begin
            Write(']');
            If ( Op_I = 'S' ) Then Write(Lst,']');
            M:=M+1;
          end
          Else
            begin
              Write('[');
              If ( Op_I = 'S' ) Then Write(Lst,'[');
              M:=M+1;
            end;
        For K:=1 To 3 Do
          begin
            Write( Linha[H] );
            If ( Op_I = 'S' ) Then Write( Lst,Linha[H] );
            H:=H+1;
            M:=M+1;
          end;
          If ( EST = 0 ) Then
            begin
              Write('[');
              If ( Op_I = 'S' ) Then Write(Lst,'[');
              M:=M+1;
            end
            Else
              begin
                Write(']');
                If ( Op_I = 'S' ) Then Write(Lst,']');
                M:=M+1;
              end;
        Write('--');
        If ( Op_I = 'S' ) Then Write(Lst,'--');
        M:=M+2;
      end;
    N:=N+1;
  Until ( N = 7 );
        Repeat
          Write(' ');
          If ( Op_I = 'S' ) Then Write(Lst,' ');
          M:=M+1;
        Until ( M >= 79 );
        Write('¦');
        If ( Op_I = 'S' ) Then WriteLn(Lst,'¦');
  end;
  Write('   ¦');
  If ( Op_I = 'S' ) Then Write(Lst,'   ¦');
```

```
    For K:=5 To 79 Do
      begin
        Write(' ');
        If ( Op_I = 'S' ) Then Write(Lst,' ');
       end;
    Write('¦');
    If ( Op_I = 'S' ) Then WriteLn(Lst,'¦');

    { ..... Mensagens de Erro ..... }
    If ( C = 'E' ) Then
      begin
        WriteLn;
        WriteLn('Erro:Sintaxe incorreta');
        Err:='S';
      end;
    If ( I >  76 ) Then
      begin
        WriteLn;
        WriteLn('Erro:Nº de caracteres excessivo');
        Err:='S';
      end;
    If ( J >= 65 ) Then
      begin
        WriteLn;
        WriteLn('Erro:Mais de 6 termos por linha');
        Err:='S';
      end;
  end;

End;     { Fim da subrotina de diagramacao e verificacao de sintaxe }



{ Bloco de leitura do relogio do microcomputador }

Procedure Tempo ( Var Si : Real );

  Type
    Registers = Record  { Variaveis para armazenar registros da UCP }
      Case Integer Of
        1 : (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags : Integer);
        2 : (AL,AH,BL,BH,CL,CH,DL,DH : Byte);
      End;

  Var
    R : Registers;
    hor,min,seg,cent : Byte;    { Dados do relogio }

  Begin

    R.AH:=$2C;
    Intr($21,R);   { Chamada de rotina de tempo do sistema operacional }
    With R Do
      begin
        hor  :=CH;
        min  :=CL;
        seg  :=DH;
        cent :=DL;
      end;
      Si:=3600*hor+60*min+seg+cent/100;  { Conversao para segundos }

  End;   { Fim bloco de leitura do relogio }
```

```
{ Bloco Interpretador de comandos para implementacao de logica }

Procedure  Logica ( Linha : Lin;
                    Var V,T,S : Vetor;
                    Var D : VetorS );

{ Declaracao de variaveis locais : }

{ H = Indice se uso geral.                                          }
{ I = Apontador para caracter de linha de instrucao.                }
{ Est1, ESt2, Est3 e Est4 = Variaveis para estados logicos.         }
{ NCOY e NESY = NCO e NES.                                          }
{ Log = Indicador de funcao.                                        }
{ Unid = Unidade de tempo programado do temporizador.               }
{ Dez = Dezena de tempo programado do temporizador.                 }
{ Sf = Tempo.                                                       }

Var
  H,I,Est1,Est2,Est3,Est4,NCOY,NESY : Byte;
          Log : Char;
  Unid,Dez,Sf : Real;


egin

 I:=1;   { Caracter inicial da linha de instrucoes. }
 If ( Linha[I] <> ' ' ) And ( Linha[I] <> ';' ) And
    ( Linha[I] <> 'F' ) Then
    begin
      Est1:=1;
      Log:=' ';
      Repeat
        If ( Linha[I] = '.' ) Then  { Teste de operacao "E". }
          begin
            Log:='E';
            I:=I+1;    { Incrementar apontador de caracter. }
          end;
        If ( Linha[I] = '(' ) Then I:=I+1;
             H:=I;
             OnOrOff ( H,EST,E,V,T,S,Linha ); { Verificar Estado. }
             Est3:=EST;
             I:=I+3;
             If ( Linha[I] = 'N' ) Then { Teste de "Negacao". }
               begin
                 I:=I+1;
                 If Est3 = 0 Then Est3:=1
                   Else Est3:=0;
               end;
             If ( Linha[I] = '.' ) Then
               begin
                 Log:='E';
                 I:=I+1;
               end;
```

```
        If ( Linha[I] = '+' ) Then  { Teste de operacao "OU". }
          begin
            Log:='O';
                H:=I+1;
                OnOrOff ( H,EST,E,V,T,S,Linha );
                Est4:=EST;
            I:=I+4;
            If ( Linha[I] = 'N' ) Then
              begin
                I:=I+1;
                If Est4 = 0 Then Est4:=1
                  Else Est4:=0;
              end;
            If ( Linha[I] = ')' ) Then I:=I+1;
          end;
        If Log = 'E' Then Est1:=Est1 And Est3; { Funcao "E". }
        If Log = 'O' Then
          begin
            Est2:=Est3 Or Est4;   { Funcao "OU". }
            Est1:=Est1 And Est2;
          end;
  Until ( Linha[I] = '=' ) Or ( I >= 75 );
          I:=I+1;
          If ( Linha[I+1] = '1' ) Then NCOY:=1;  { Caracter para }
          If ( Linha[I+1] = '2' ) Then NCOY:=2;  { numeros.      }
          If ( Linha[I+1] = '3' ) Then NCOY:=3;
          If ( Linha[I+1] = '4' ) Then NCOY:=4;
          If ( Linha[I+1] = '5' ) Then NCOY:=5;
          If ( Linha[I+1] = '6' ) Then NCOY:=6;
          If ( Linha[I+1] = '7' ) Then NCOY:=7;
          If ( Linha[I+1] = '8' ) Then NCOY:=8;
          If ( Linha[I+2] = '1' ) Then NESY:=1;
          If ( Linha[I+2] = '2' ) Then NESY:=2;
          If ( Linha[I+2] = '3' ) Then NESY:=3;
          If ( Linha[I+2] = '4' ) Then NESY:=4;
          If ( Linha[I+2] = '5' ) Then NESY:=5;
          If ( Linha[I+2] = '6' ) Then NESY:=6;
          If ( Linha[I+2] = '7' ) Then NESY:=7;
          If ( Linha[I+2] = '8' ) Then NESY:=8;
          If ( Linha[I] = 'V' ) Then V[NCOY,NESY]:=Est1; { Atualizar }
          If ( Linha[I] = 'S' ) Then S[NCOY,NESY]:=Est1; { valores.  }
          If ( Linha[I] = 'T' ) Then
            begin

              { ..... Teste de Temporizador Operado ..... }

              If ( Est1 = 0 ) Then
                begin
                  T[NCOY,NESY]:=0;
                  D[NCOY,NESY]:=0;
                end;
              If ( Est1 = 1 ) Then
                begin
                  If ( D[NCOY,NESY] <= 0.01 ) Then
                    begin
                      Tempo ( Si );     { Leitura de tempo. }
                      D[NCOY,NESY]:=Si; { Armazenar.        }
                    end
                    Else
```

```
                        begin
                          Tempo ( Si );
                          Sf:=Si-D[NCOY,NESY]; { Tempo decorrido. }

                          { Conversao de caracteres em numeros. }
                          If ( Linha[I+5] = '0' ) Then Unid:=0;
                          If ( Linha[I+5] = '1' ) Then Unid:=1;
                          If ( Linha[I+5] = '2' ) Then Unid:=2;
                          If ( Linha[I+5] = '3' ) Then Unid:=3;
                          If ( Linha[I+5] = '4' ) Then Unid:=4;
                          If ( Linha[I+5] = '5' ) Then Unid:=5;
                          If ( Linha[I+5] = '6' ) Then Unid:=6;
                          If ( Linha[I+5] = '7' ) Then Unid:=7;
                          If ( Linha[I+5] = '8' ) Then Unid:=8;
                          If ( Linha[I+5] = '9' ) Then Unid:=9;
                          If ( Linha[I+4] = '0' ) Then  Dez:=0;
                          If ( Linha[I+4] = '1' ) Then  Dez:=1;
                          If ( Linha[I+4] = '2' ) Then  Dez:=2;
                          If ( Linha[I+4] = '3' ) Then  Dez:=3;
                          If ( Linha[I+4] = '4' ) Then  Dez:=4;
                          If ( Linha[I+4] = '5' ) Then  Dez:=5;
                          If ( Linha[I+4] = '6' ) Then  Dez:=6;
                          If ( Linha[I+4] = '7' ) Then  Dez:=7;
                          If ( Linha[I+4] = '8' ) Then  Dez:=8;
                          If ( Linha[I+4] = '9' ) Then  Dez:=9;
                          Dez:=10*Dez+Unid;

                          { Verificacao de tempo de operacao. }
                          If ( Sf >= Dez ) Then T[NCOY,NESY]:=1;
                        end;
                    end;
                end;
    end;

End;        { Final da subrotina de implementacao de logica }
```

```
{ Subrotina de Edicao das listas de instrucoes          }

Procedure Editor ( Var  LI : Byte;   { Bloco de Edicao }
                   Var Linha : Lin;
                   Var Texto : Text;
                   Var Op_I : Char );

  Type Prog = Record Linh : String [75];
                     end;
  A = File Of Prog;

  { Declaracao de variaveis locais :  }

  { Flag = Bandeira de indicacao de erro em leitura de arquivo de disco. }
  { I = Indice de uso geral.                                             }
  { LF = Linha final.                                                    }
  { Nom_arq = Nome de arquivo para armazenar programas em disco.         }

  Var Flag : Boolean;
      I , LF : Byte;
      Nom_arq : String [8];
      Progr : Prog;
      Arq : A;

  Begin

    ClrScr;
    GoToXY(32,4);
    Write('EDITOR');
    GoToXY(14,6);
    Write('[ Veja  as regras de sintaxe no manual de utilizacao ]');
    GoToXY(18,10);
    Write('- Limpar Buffer de Edicao, pressione <S> ou <N> :');
    Repeat
      Teclado ( tec );  { Acionar subrotina de leitura de teclado }
    Until (tec='S') Or (tec='N'); { Repetir ate ocorrer teste }
    If (tec='S') Then
      begin
        For I:=1 To 200 Do
        Texto[I]:='                                                    ';
      end;
    GoToXY(18,14);
    Write('- Carregar programa de disco, pressione <S/N> :');
    Repeat
      Teclado ( tec );  { Acionar subrotina de leitura de teclado }
    Until (tec='S') Or (tec='N'); { Repetir ate ocorrer teste }
    If (tec='S') Then
      begin
        WriteLn;WriteLn;
        Repeat
          Write('=> Digite o nome do arq. (8 carac.) e press. <ENTER> : ');
          Read(Nom_arq);
          Assign(Arq,Nom_arq);      { Associacao do nome ao arquivo }
          {$I-} Reset(Arq) {$I+};   { Abrir arquivo }
          Flag := (IOresult = 0);         { Teste de erro de I/O}
          If Not Flag Then                { de arquivo nao existente}
```

```
      begin
        WriteLn;WriteLn;
        WriteLn('  Arquivo inexistente');
        WriteLn;
        Write('  Pressione uma tecla para outro nome,se nao <N>');
        Read(Kbd,tec);
        tec:=UpCase(tec);
        WriteLn;WriteLn;
      end;
  Until Flag Or (tec='N');
  If Flag Then
    begin
      Read(Arq,Progr);  { Leitura de arquivo }
      I:=1;
      Texto[I]:=Progr.Linh;
      While ( Not Eof(Arq)) Do   { Teste final de arquivo }
        begin
          Read(Arq,Progr);  { Leitura arquivo e armazenamento}
          I:=I+1;                    { na matriz texto       }
          Texto[I]:=Progr.Linh;
        end;
      Close(Arq);             { Fechar Arquivo }
    end
    Else
      begin
        Close(Arq);           { Fechar Arquivo }
      end;
end;

Tela_Op_Ed;    { Bloco Tela Opcoes do Editor }

Repeat
  Teclado ( tec );
  If (tec='O') Then Tela_Op_Ed;

  { ............. Rotina de Digitacao.............. }
  If (tec='D') Then
    begin
      Num_Linha ( LI );
      WriteLn;WriteLn;
      WriteLn('  Digite as Instrucoes ( ate a marca limitadora = ! )');
      WriteLn('  por linha e pressione <ENTER>');
      WriteLn;WriteLn;
      WriteLn('  Para deixar Insercao digite espaco em branco');
      WriteLn('  e <ENTER> na linha posterior a ultima linha digitada');
      WriteLn;
      Repeat
        For I:=1 To 78 Do
          Write(' ');
        WriteLn('!');
        If ( LI <  10 ) Then Write('  ',LI,' ');
        If ( LI >= 10 ) And ( LI <= 99 )  Then Write(' ',LI,' ');
        If ( LI >=100 ) And ( LI <=200 ) Then Write(LI,' ');
        Linha:='                                                  ';
        ReadLn ( Linha );
        Texto[LI]:=Linha;
        LI := LI + 1;
        WriteLn;
      Until ( LI > 200 ) Or ( Linha = ' ' );
      WriteLn;WriteLn;
    end;
  { ......... Final Rotina de Digitacao ......... }
```

```
{ ..... Rotina Inserir Linha ..... }

If ( tec='I' ) Then
  begin
    Num_Linha ( LI );
    LF:=200;
    Repeat
      Texto[LF]:=Texto[LF-1];
      LF:=LF-1;
    Until ( LF = LI );
    For I:=1 To 78 Do
      Write(' ');
    WriteLn('!');
    If ( LI <  10 ) Then Write('  ',LI,' ');
    If ( LI >= 10 ) And ( LI <= 99 ) Then Write(' ',LI,' ');
    If ( LI >= 99 ) And ( LI <=200 ) Then Write(LI,' ');
    Linha:='                                                                ';
    ReadLn ( Linha );
    Texto[LI]:=Linha;
    WriteLn;WriteLn;
  end;

{ ..... Final Rotina Inserir Linha ..... }



{ ..... Rotina Apagar Linha ..... }

If ( tec = 'A' ) Then
  begin
    Num_Linha ( LI );
    LF:=200;
    Repeat
      Texto[LI]:=Texto[LI+1];
      LI:=LI+1;
    Until ( LI = LF );
    Texto[LF]:=' ';
    WriteLn;
  end;

{ ..... Final Rotina Apagar Linha ..... }


{ ..... Rotina de Listagem ..... }

If ( tec = 'L' ) Then
  begin
    WriteLn;WriteLn;
    Repeat
      Write('  Digite a Linha Inicial e pressione <ENTER> :');
      Read(LI);
      WriteLn;WriteLn;
      Write('  Digite a Linha Final e pressione  <ENTER>  :');
      Read(LF);
      WriteLn;WriteLn;
    Until ( LI>=1 ) And ( LF<=200 ) And ( LI<=LF );
    Write('  <I>=Impressora / <QUALQUER OUTRA TECLA>=Video :');
    Teclado ( tec );
    If tec='I' Then
```

```
          begin
            WriteLn;WriteLn;
            Write('  Ligue a Impressora e depois pressione uma tecla');
            Read(Kbd,tec);
            WriteLn;
            WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);
            Repeat
            Write(Lst,LI);
            Write(Lst,' ');
            Write(Lst,Texto[LI]);
            WriteLn(Lst);WriteLn(Lst);
            LI:=LI+1;
            Until LI>LF;
          end
          Else
            begin
              WriteLn;WriteLn;
              Repeat
                Write(LI);
                Write(' ');
                Write(Texto[LI]);
                WriteLn;WriteLn;
                LI:=LI+1;
              Until ( LI>LF );
            end;
      end;

  { ..... Final Rotina de Listagem ..... }



  { ..... Rotina de Gravacao ..... }

  If ( tec='G' ) Then
    begin
      WriteLn;WriteLn;
      Write('=> Digite o nome do arq. (8 carac.) e press. <ENTER> :');
      Read(Nom_arq);
      Assign(Arq,Nom_arq);  { Associacao do nome ao arquivo }
      Rewrite(Arq);           { Abrir arquivo }
      I:=1;
      Progr.Linh:=Texto[I];
      Repeat
        Write(Arq,Progr); { Escrever no arquivo }
        I:=I+1;
        Progr.Linh:=Texto[I];
      Until ( I = 200 ) Or ( Texto[I]='F' );
      Write(Arq,Progr);   { Escrever no arquivo }
      Close(Arq);  { Fechar arquivo }
      WriteLn;WriteLn;
      WriteLn(' Arquivo Gravado');
      WriteLn;
    end;

{ ..... Final Rotina de Gravacao ..... }
```

```
    { ..... Rotina Sintaxe e Diagrama de Rele ..... }

    If ( tec = 'R' ) Then
      begin

        WriteLn;WriteLn;
        Write('=> Pressione <I> para Impressora se nao outra tecla');
        Teclado ( tec );
        WriteLn;
        If ( tec = 'I' ) Then
          begin
            Op_I:='S';
            Write('=> Ligue a Impressora e depois pressione uma tecla');
            Teclado ( tec );
            WriteLn;
            WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);
          end;
        Num_Linha ( LI );
        Linha:=Texto[LI];
        WriteLn;WriteLn;
        If ( LI <  10  ) Then Write('  ',LI,' ',Linha);
        If ( LI >= 10  ) And ( LI <= 99  ) Then Write(' ',LI,' ',Linha);
        If ( LI >= 100 ) And ( LI <= 200 ) Then Write(   LI,' ',Linha);
        WriteLn;WriteLn;
        Logica ( Linha,V,T,S,D );        { Subrotina de Logica.      }
        Sint_Rele ( LI,Linha,Err,Op_I ); { Subrotina de Diagramacao. }
        WriteLn;WriteLn;

      end;

    { .. Final Rotina Sintaxe e Diagrama de Rele .. }

  Until ( tec='M');

End; { Fim Bloco de Edicao }



{ Este bloco realiza as simulacoes na forma de diagrama de contatos }

Procedure Simulador ( Var LI : Byte;
                      Linha : Lin;
                      Texto : Text;
                      Var E,V,T,S : Vetor;
                      Var Op_I : Char );

  Begin

    ClrScr;
    WriteLn;WriteLn;
    Write('=> Pressione <I> para Impressora se nao outra tecla');
    Teclado ( tec );
    WriteLn;
    If ( tec = 'I' ) Then
      begin
        Op_I:='S';
        Write('=> Ligue a Impressora e depois pressione uma tecla');
        Teclado ( tec );
        WriteLn;
        WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);
      end;
```

```
    Repeat
      WriteLn;
      Status ( E );  { Le Entradas pelo teclado. }
      LI:=1;
      Linha:=Texto[LI]; { Obtem linha do texto. }
      { Varredura das linhas de instrucoes de um programa editado(texto). }
      While ( LI <= 200 ) And ( Linha <> 'F' ) And ( Err <> 'S' ) Do
        begin
          Logica (Linha,V,T,S,D); { Implementacao da logica.    }
          LI:=LI+1;               { Proxima linha de instrucao. }
          Linha:=Texto[LI];
        end;
      LI:=1;
      Linha:=Texto[LI];
      While ( LI <= 200 ) And ( Linha <> 'F' ) And ( Err <> 'S' ) Do
        begin
          Sint_Rele ( LI,Linha,Err,Op_I ); { Subrotina de Diagramacao. }
          LI:=LI+1;
          Linha:=Texto[LI];
        end;
      WriteLn;
      If ( Err = 'S' ) Then WriteLn('Erro no programa');
      Write('=>Pressione <P>=Parar, outra <Tecla>=Continuar Simulacao');
      Teclado ( tec );
      WriteLn;WriteLn;
      If ( Op_I = 'S' ) Then
        begin
          WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);WriteLn(Lst);
        end;
    Until ( tec = 'P' );
    WriteLn;WriteLn;

  End;    { Fim bloco Simulador }



{ Subrotina de verificacao de sintaxe para Bloco de Interface }
{ Listagem semelhante ao bloco Sint_Rele sem instrucoes de impressao. }

Procedure Verif_Sint ( LI : Byte;
                       Linha : Lin;
                       Var Err : Char);

Type  Num = '0'..'9';
      Cod = Set Of Num;

Var
   H,I,M : Byte;
         C : Char;
     Temp,Codg : Cod;


Begin

  Err:='N';
  Temp:=['0'..'9'];
  Codg:=Temp - ['0','9'];
  I:=1;
  If ( Linha[I] <> ' ' ) And ( Linha[I] <> ';' ) And
     ( Linha[I] <> 'F' ) Then
```

```
begin
  C:=' ';
  Repeat
    If ( Linha[I] = '.' ) Then I:=I+1;
    If ( Linha[I] = '(' ) Then I:=I+1;
    If ( Linha[I] <> 'E' ) And ( Linha[I] <> 'V' ) And
       ( Linha[I] <> 'S' ) And ( Linha[I] <> 'T' ) Then C:='E'
     Else
       begin
         If ( Linha[I+1] In Codg ) Then
           Else C:='E';
         If ( Linha[I+2] In Codg ) Then
           Else C:='E';
             I:=I+3;
         If ( Linha[I] = 'N' ) Then I:=I+1;
         If ( Linha[I] = '.' ) Then I:=I+1;
         If ( Linha[I] = '+' ) Then
           begin
             If ( Linha[I+1] = 'E' ) Or ( Linha[I+1] = 'S' ) Or
                ( Linha[I+1] = 'V' ) Or ( Linha[I+1] = 'T' ) Then
               begin
                 If ( Linha[I+2] In Codg ) Then
                   Else C:='E';
                 If ( Linha[I+3] In Codg ) Then
                   Else C:='E';
               end
               Else C:='E';
             I:=I+4;
             If ( Linha[I] = 'N' ) Then I:=I+1;
             If ( Linha[I] = ')' ) Then I:=I+1
               Else C:='E';
           end;
       end;
  Until ( Linha[I] = '=' ) Or ( I >= 75 ) Or  ( C = 'E' );

  If ( Linha[I] <> '=' ) Or ( I >= 75 ) Then C:='E'
    Else
      begin
        I:=I+1;
        If ( Linha[I] = 'S' ) Or ( Linha[I] = 'V' ) Or
           ( Linha[I] = 'T' ) Then
          begin
            If ( Linha[I+1] In Codg ) Then
              Else C:='E';
            If ( Linha[I+2] In Codg ) Then
              Else C:='E';
          end
          Else C:='E';
        If ( Linha[I] = 'T' ) Then
          begin
            If ( Linha[I+4] In Temp ) Then
              Else C:='E';
            If ( Linha[I+5] In Temp ) Then
              Else C:='E';
          end;
            I:=I+3;
      end;
```

```
      If ( C = 'E' ) Then
        begin
          WriteLn;
          WriteLn('Erro:Sintaxe incorreta');
          Err:='S';
        end;
      If ( I >  76 ) Then
        begin
          WriteLn;
          WriteLn('Erro:Nº de caracteres excessivo');
          Err:='S';
        end;
    end;

End;      { Final subrotina Verif_Sint }




{ Bloco para leitura das entradas digitais do interface }

Procedure Le_Entrada ( Var E : Vetor );

  Const ES1 = $220;  { Endereco da porta de entrada. }

  { Declaracao de variaveis locais : }

  { D = Dado lido da porta de entrada.                    }
  { R e V = Variaveis.                                    }
  { I = Indice de uso geral.                              }

  Var D,R,V : Byte;
        I : Byte;

  Begin

    D:=Port[ES1]; { Instrucao de leitura da porta de entrada do interface. }
    R:=1;
    For I:=1 To 8 Do
      begin
        V:=D And R;
        If V=0 Then E[1,I]:=0   { Armazena na matriz }
          Else E[1,I]:=1;       { de entradas.       }
        R:=R Shl 1;
      end;

  End;     { Final bloco de leitura de interface }
```

```
{ Bloco para escrita nas saidas digitais do interface }

Procedure Atua_Saida ( S : Vetor );

  Const ES1 = $220;  { Endereco da porta de saida. }

  Var D : Byte;
      I : Byte;

  Begin

    D:=0;
    For I:=1 To 8 Do
      D:=D Or ( S[1,I] Shl (I-1) );
    Port[ES1]:=D; { Instrucao de escrita da porta de saida do interface. }

  End;   { Final bloco escrita em interface }




{ Subrotina para desativar saidas do interface }

Procedure Desat_Saida;

  Const ES1 = $220;  { Endereco da porta de saida. }

  Begin

    Port[ES1]:=0;    { Nivel logico baixo nas portas de saida. }

  End;   { Fim subrotina }
```

```
{ Bloco de Simulacao utilizando o interface para controles em tempo real }

Procedure Interface ( Var LI : Byte;
                      Linha : Lin;
                      Texto : Text;
                      Var E,V,T,S : Vetor;
                      .       Err : Char );
  Var tc : Byte;  { Variavel para leitura de teclado. }

  Begin

    ClrScr;
    WriteLn('..... Teste Sintaxe Programa .....');
    LI:=1;
    Linha:=Texto[LI];
    { Varredura das linhas de instrucoes. }
    While ( LI <= 200 ) And ( Linha <> 'F' ) And ( Err <> 'S' ) Do
      begin
        Verif_Sint ( LI,Linha,Err ); { Verificacao de sintaxe }
        LI:=LI+1;
      end;
    WriteLn;WriteLn;
    If Err = 'S' Then
      begin
        WriteLn('=> Erro de Sintaxe no Programa');
        WriteLn;WriteLn;
      end
      Else
        begin
          WriteLn('..... Programa sem Erros de Sintaxe .....');
          WriteLn;WriteLn;
          WriteLn('=> Pressione uma tecla para comecar execucao');
          WriteLn('  Para bloquear a execucao pressione a tecla <B>');
          Teclado ( tec );
          WriteLn;WriteLn;
          WriteLn('..... Execucao Iniciada ..... ');
          Mem[0000:1054]:=0;  { Limpar buffer teclado }
          Repeat
            Le_Entrada ( E ); { Leitura das entradas digitais. }
            LI:=1;
            Linha:=Texto[LI];
            While ( LI <= 200 ) And ( Linha <> 'F' ) Do
              begin
                Logica ( Linha,V,T,S,D ); { Implementacao da Logica. }
                LI:=LI+1;
                Linha:=Texto[LI];
              end;
            Atua_Saida ( S ); { Ativacao das saidas digitais. }
            tc:=Mem[0000:1054];  { Ler buffer teclado }
            If ( tc <> 0 ) Then Mem[0000:1054]:=0; { Limpar buffer teclado }
          Until ( tc = 98 ) Or ( tc = 66 ); { Repetir ate pressionar <B> }
          Desat_Saida;    { Saidas digitais em nivel baixo. }
          WriteLn;WriteLn;
          WriteLn('..... Execucao Interrompida ..... ');
        end;

End;   { Final Bloco de Simulacao com interface de entrada e saida }
```

```
Procedure Tela_Menu;  { Subprograma Tela_Menu }

  Begin

    ClrScr; { Limpar tela }
    GoToXY(36,2); { Posicionar cursor }
    Write('MENU'); { Impresao Menu }
    GoToXY(36,3);
    Write('----');
    GoToXY(22,6);
    Write('- Editor <E>'); { Opcoes }
    GoToXY(22,8);
    Write('- Simulacao no video ou na impressora <S>');
    GoToXY(22,10);
    Write('- Simulacao com Interface <I>');
    GoToXY(22,12);
    Write('- Retorno ao Menu <M>');
    GoToXY(22,14);
    WriteLn('- Finalizar Sessao <F>');

  End;  { Fim Tela_Menu }




BEGIN { Bloco Principal }

  ClrScr;                          { Limpar tela }
  GoToXY(30,5);                    { Posicionar cursor }
  Write('PROGRAMA  SIMULADOR  DE'); { Apresentacao }
  GoToXY(20,10);
  Write('  CONTROLADORES  LOGICOS  PROGRAMAVEIS');
  GoToXY(20,15);
  WriteLn('Projetista : Carlos Alberto Murari Pinheiro / EFEI-DON');
  GoToXY(20,20);
  WriteLn('Pressione qualquer tecla para continuar');
  Read(Kbd,tecla);
  Tela_Menu; { Bloco Menu }
  Repeat
    For LI:=1 To 8 Do
      begin
        For EST:=1 To 8 Do
          begin
            E[LI,EST]:=0;   { Estado inicial das matrizes. }
            V[LI,EST]:=0;
            T[LI,EST]:=0;
            S[LI,EST]:=0;
            D[LI,EST]:=0;
          end;
      end;
    Read(Kbd,tecla); { Leitura do Teclado }
    tecla:=UpCase(tecla);
    Op_I:='N';
    { Modalidades do Simulador de CLP }
    If  (tecla='E') Then Editor ( LI,Linha,Texto,Op_I );
    If  (tecla='S') Then Simulador ( LI,Linha,Texto,E,V,T,S,Op_I );
    If  (tecla='I') Then Interface ( LI,Linha,Texto,E,V,T,S,Err );
    If  (tecla='M') Then Tela_Menu;
  Until (tecla='F');

END. { Fim Bloco Principal }

{          FIM DO PROGRAMA SIMULADOR DE CONTROLADOR PROGRAMAVEL          }
```

REFERÊNCIAS BIBLIOGRÁFICAS

- Flaherty, Michael R. ; Increased capabilities of programmable controllers; IEEE IAS V1 p 352-357 1980. [2]

- Linnenbrink, T. E. ; Microprocessor based control; IEEE IECI V23 N3 p 230-232 Aug 1976. [5]

- Maggioli, Victor J. ; Programmable controllers in chemical process applications; IEEE IA V15 N6 p 695-700 Nov/Dec 79. [4]

- Michalsky, G. E. ; Performing machine diagnostics with programmable controllers; IEEE IA V20 N5 p 1308-1311 Sep/ oct 1984. [3]

- Palau, Yves G. ; L´Automate programmable, Educalivre - Paris p 98 1983. [7]

- Tuttle, William H. ; Systems incorporating programmable controllers; IEEE Conf. Eletrc. Prob. p 32-38 Apr. 1980. [1]

- Zsombor-Murray, P. J. A. ; Binary decision based programmable controllers; IEEE Micro V3 N6 p 24-39 Dec. 1983. [6]

- Manuais de Fabricantes : [8]
  - Maxitec SA; Controlador Programável MXT 110.
  - Metal Leve Controles Eletrônicos LTDA; Controlador Programável CLP 5.
  - Pulse Tecnologia Digital Industria Eletrônica LTDA, Minicontrolador programável CLP-40.
  - Siemens SA, Aparelho de Automatização SIMATIC S5-110.

DATA 22/08/1989
PROC
PED. P.P.G.
LIV.
Cz$ DOACAO

I.V N T O
BIM - EFEI
DATA | Publ a

EFEI - BIBLIOTECA MAUÁ
8200586
NÃO DANIFIQUE ESTA ETIQUETA