Universidade de Évora

Mestrado de Energia e Ambiente

Modelização de Sistemas de Gestão de Energia em Ambiente

Realizado

Teresa Deveza nº17078

Orientado

Prof. João Martins

Esta dissertação não inclui as críticas e sugestões feitas pelo júri.

Universidade de Évora

Mestrado de Energia e Ambiente

Modelização de Sistemas de Gestão de Energia em Ambiente

Realizado

Teresa Deveza nº17078

Orientado

Prof. João Martins

Esta dissertação não inclui as críticas e sugestões feitas pelo júri.

Índice

Resumo

Este trabalho propõe uma metodologia de tradução para igualar o programa de controle de PLC no ambiente Matlab/Simulink. A lista traduz automaticamente o programa de controlo de PLC para a linguagem de software Matlab/Simulink. O programa do PlC é traduzido para uma função bloco do Matlab, dentro do ambiente Matlab/Simulink, que irá controlar o modelo do processo industrial, desde que a simulação seja executada. As entradas e saídas da lista de tradução do PLC depende do tipo de autómato que é escolhido. A lista de tradução será compatível com um ficheiro Matlab/Simulink que corresponde tradução de programa de controle de PLC.

Abstract

Modulation of Systems of Management of Energy in Environment

This work proposes a translation methodology to equal the program of control of PLC in the environment Matlab/Simulink. The list translates automatically the program of control of PLC for the language of software Matlab/Simulink. The program of the PlC is translated for a function block of the Matlab, inside the environment Matlab/Simulink, which will be going to control the model of the industrial process, since the simulation is executed. The entries and exits of the translation list of the PLC it depends on the type of automaton that is chosen. The translation list will be compatible with a filing cabinet Matlab/Simulink that corresponds translation of program of control of PLC.

1. Objectivo

Em meios industriais e académicos MATLAB/SIMULINK é utilizado como software elevado desempenho direccionado para o cálculo numérico e simulação dos sistemas. O MATLAB/SIMULINK permite a realização de aplicações ao nível da análise numérica, de análise de dados, cálculo matricial, processamento de sinais e construção de gráficos, entre outras, abrangendo um leque alargado de problemas científicos e de engenharia.

O objectivo do presente trabalho é desenvolver um tradutor da linguagem do autómato para linguagem MATLAB/SIMULINK. Desta forma, o modelo do processo industrial é simulado em ambiente MATLAB/SIMULINK, sendo possível verificar o desempenho do programa desenvolvido por autómato programável face ao processo que se pretende controlar.

2. Introdução

O processo de controlo e automação é comandado por controladores lógicos programáveis (PLC) é um problema bem conhecido. Há várias soluções que podem ser implementadas: modelos de escale, as baterias de led's, comutadores Ser Humano Trabalham a Máquina Interfaces (HMI), aquisição de Dados e Controle de Supervisão (SCADA) instrumentos de simulação ou sistemas. O uso de modelos de escala de verdadeiros processos é muito caro e difícil de adaptar-se a processos diferentes. Não há dúvida que este é o melhor modo de ensinar a PLC, permitindo aos estudantes testar os seus projectos em um ambiente quase verdadeiro, contudo o seu preço muitas vezes proíbe o seu uso. O uso de led's e jogos de comutadores é o fim extremamente confuso e desinteressante. Esta abordagem é válida apenas quando os processos são considerados pequenos, reduz fortemente a motivação do formando. Os HMI e os sistemas SCADA permitem esta característica mas são muito caros, não destinados a esta finalidade e normalmente consideram a propriedade de protocolos. Alguns instrumentos de simulação de processo baseiam-se nos PC que foram desenvolvidos, usando tecnologias de microcontrolador e projectado para trabalhar com qualquer tipo de PLC. Outros instrumentos de simulação PLC comerciais são também disponíveis (PC-SIM, SIMTSX, PSIM somente para citar alguns). Contudo, muitas vezes essas soluções não são convenientes para estar integrado a outros instrumentos de simulação.

O uso de Matlab/Simulink não é uma aproximação regular para ensinar a automação industrial e PLC. A finalidade do modelo do processo industrial a ser implementado no Matlab/Simulink sendo este eficaz a implementar um programa de controlo PLC. A ideia básica deve considerar o programa de controle de PLC como um bloco de função do1 Matlab, dentro do ambiente Matlab/Simulink, que controlará o modelo do processo industrial enquanto a simulação decorre. O objectivo principal deste trabalho é traduzir automaticamente o programa de controle de PLC, escrito como uma lista de instrução, na língua de software Matlab/Simulink.

3. Estudo da arte

Existem vários tipos de Software para simular a programação de autómatos. Um deles é o software SIMTSX. Este programa destina-se à afinação de programas para autómatos Micro/Premium/Quantum, sem ligação à máquina nem ao processo. Este software reproduz num micro-computador, PC, o comportamento da máquina ou do processo a afinar.

As vantagens da utilização do software SIMTSX são as seguintes:

- Colocação em serviço facilitada e mais rápida (montagem em cadência acelerada).
- Maior segurança dos meios de produção (disponibilidade, segurança).
- Supressão quase total das perdas de produção devidas aos disfuncionamentos do software.
- Formação mais rápida dos operadores.
- Redução das intervenções sob garantia (em número de ocorrências e em gravidade).

O software SIMTSX permite simular o comportamento dos sensores (ligados às entradas do autómato) e dos accionadores (ligados às saídas do autómato), simulando assim o funcionamento completo de uma máquina ou de um processo.

O software SIMTSX pode reproduzir o comportamento de uma máquina ou de um processo controlado por 8 autómatos no máximo, ligados por uma rede de comunicação.

Um outro exemplo é o JFLAP (Java Formal Language and Automata Package) é uma ferramenta visual usada para criar e simular diversos tipos de autômatos, e converter diferentes representações de linguagens. Ele pode ser usado tanto como auxílio às aulas, como ferramenta de estudo e pesquisa, facilitando a criação de autômatos, quanto a verificação se estão corretos.

O passo inicial para a utilização do JFLAP é a criação de autômatos. O programador utiliza a interface visual para criar um grafo representando os estados, o diagrama de transições e os seus labels. Em seguida, pode definir a palavra de entrada e então visualiza a execução de cada passo do autômato, verificando se o seu projeto é

coerente. Isso pode ser feito com três escolhas de execução, um modo rápido, que indica a resposta imediata, um modo passo a passo, que mostra os estados percorridos, e o modo múltiplo, que mostra o teste de diversas palavras, da mesma maneira que o modo rápido.

O PSIM é uma lista de simulação projecta especificamente para a electrónica e controle motor. A lista de simulação PSIM compõe-se de três programas: circuito editor esquemático SIMCAD *, simulador de PSIM, e o programa SIMVIEW* de processamento de forma de onda. A simulação o ambiente e é ilustrado como se segue.

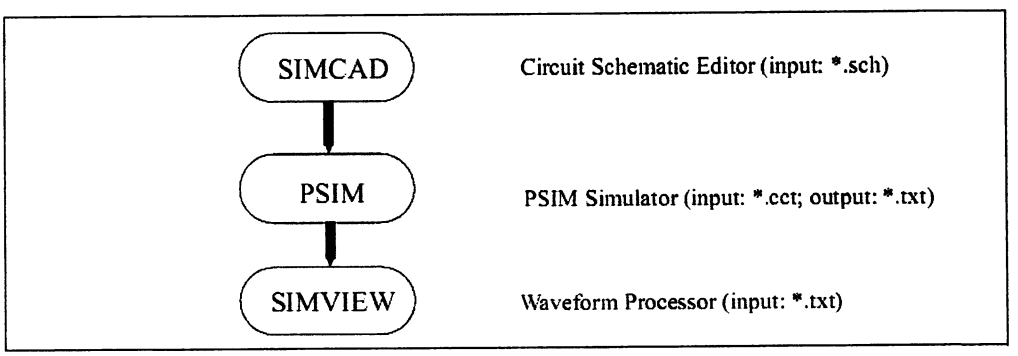


Fig.1 – Esquema da composição do simlador PSIM.

Um circuito é representado em PSIM em quatro blocos: circuito de poder, circuito de controle, sensores, e controladores de comutador. A figura abaixo de demonstrações a relação entre esses blocos

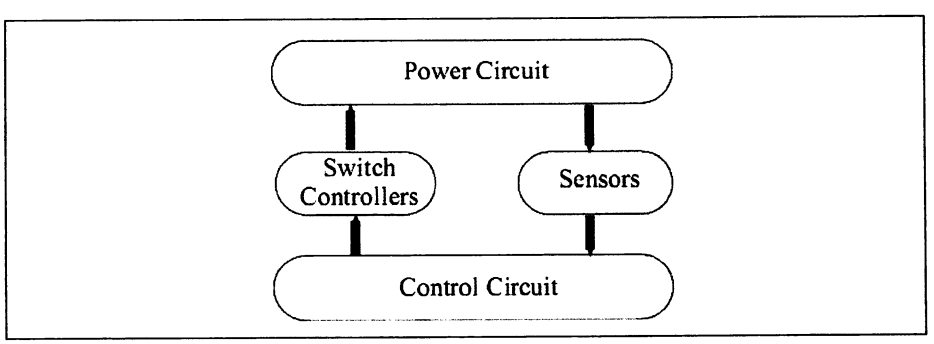


Fig.2 – Esquema do circuito do Simulador PSIM.

O circuito de poder compõe-se de trocar dispositivos, ramos de RLC, transformadores, e indutores ligados. O circuito de controle é representado no diagrama de bloco. Os componentes em domínio s e domínio z, componentes lógicos (como portas lógicas), e componentes não lineares (como multiplicadores e divisores) podem ser usados no circuito de controle. Os sensores medem voltagens de circuito de poder e

correntes e passo os valores ao circuito de controle. Os sinais de passagem então são gerados do circuito de controle e enviados atrás ao circuito de poder por controladores de comutador para controlar comutadores.

O Sistema de Supervisão e Aquisição de Dados é abreviada SCADA (Supervisory Control and Data Aquisition) são sistemas que utilizam software para monitorar e supervisionar as variáveis e os dispositivos de sistemas de controle conectados através de drivers específicos. Um dos principais processos de SCADA é a capacidade de controlar todo um sistema em tempo real. Isto é facilitado pela aquisição de dados, incluindo leitura de medidor, verificação de status de sensores, que são transmitidos em intervalos regulares, dependendo do sistema.

Os sistemas SCADA utilizam tecnologias de computação e comunicação para automatizar a monitorização e controlo dos processos industriais, efectuando recolha de dados em ambientes complexos, eventualmente dispersos geograficamente, e a respectiva apresentação de modo amigável para o utilizador, com recurso a interfaces Homém-Máquina.

O programa SOLPICA (Simulador ON-Line de Processos Industriais Controlados por Autómatos), foi desenvolvido na Escola Superior Tecnologia de Setúbal, consiste em simular processos industriais num programa desenvolvido em Visual Basic C++.

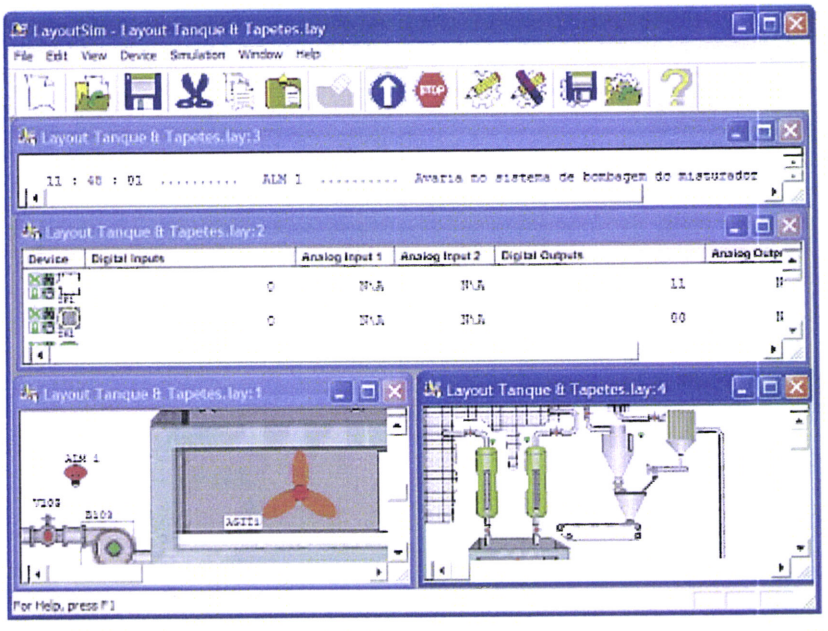


Fig. 3 - O resumo de um PLC simulado controlou o processo industrial

Contendo uma consola desenvolvida pelo mesmo, que fará a ligação entre a simulação do processo industrial e o PLC, onde será programado.

Fig.4 – Interface entre o computador e o PLC.

A desvantagem deste projecto é o preço da consola que é muito elevado.

4. Autómato programável

PLC, do inglês Programmable Logic Controller, ou CLP, Controlador Lógico Programável, significa que é um equipamento electrónico digital, cujo hardware e software são compatíveis com as aplicações industriais. Este equipamento contem uma memória programável que armazena internamente instruções e funções especificas, tais como, lógica, sequencialmente, temporização, contagem e aritmética, controlando, por meio de módulos de entradas e saídas, vários tipos de máquinas ou processos.

PLC pode ser entendido como um equipamento constituído por *Hardware* e *Software* especificamente projectados para aplicações de controlo de variáveis através de módulos de entradas e saídas digitais e analógicas.

O programa de controlo de PLC é cíclico executado do modo fluente: a unidade de controlo central copia o estado do processo industrial (circuitos de entrada de PLC) na área de memória de trabalho interna, logo executa o programa de controlo de PLC fornecido na área de memória de programa de controlo, e finalmente actua por cima do processo industrial transmitindo as suas acções de controlo pelos circuitos de produção aos accionadores de processo industriais. Esta execução é feita ciclicamente, como O programa de controlo de PLC gerará produções que serão as entradas do processo industrial, como as produções do processo industrial serão as entradas do programa de controlo de PLC.

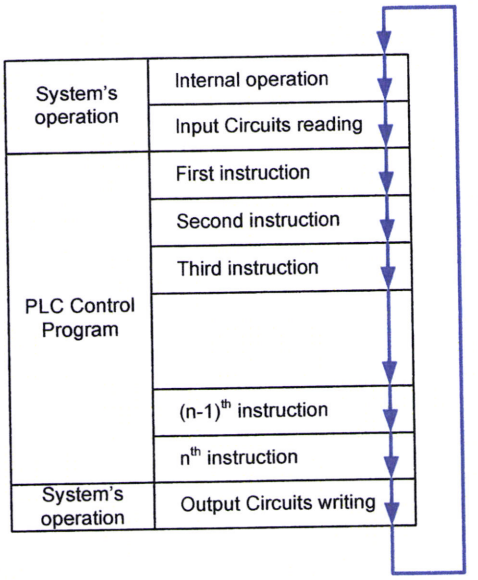


Fig.5 – Ciclo de execução do programa de controlo PLC.

No início de cada ciclo de execução as entradas (input), que podem derivar de um qualquer processo, são analisadas. No final de cada ciclo as saídas são actuadas, de acordo com as entradas e o programa de controlo, de modo a comandar o equipamento ou máquinas envolvidas no processo. Na execução do programa, o CPU utiliza a memória do autómato para armazenamento e transferência de dados. A utilização da memória para armazenamento do programa, separada da outra memória, tem a ver com a necessidade de armazenar os programas no PLC, mesmo com a alimentação desligada.

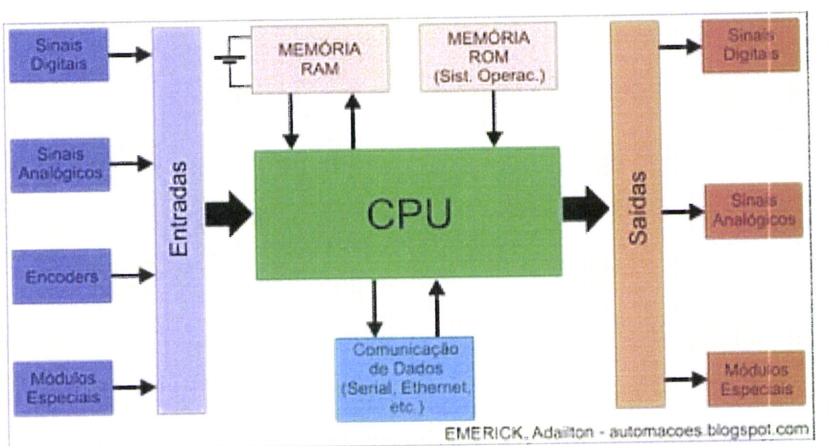


Fig. 6 – Estrutura simplificada em blocos de um PLC

O sistema de entradas/saídas é um dos componentes mais importantes num autómato pois estas necessitam de interagir directamente com equipamento industrial.

Geralmente os autómatos utilizam memória do tipo RAM, EPROM ou EEPROM. Na maioria dos casos a memória do tipo RAM é utilizada nas fases de desenvolvimento e teste dos programas, enquanto as memórias do tipo EPROM e EEPROM são utilizadas para o armazenamento de programas em código executável e também para armazenamento de configurações do sistema. No entanto, hoje em dia, a tendência é para a utilização de memória RAM, devido ao seu baixo consumo, juntamente com baterias que permitem manter o conteúdo da memória mesmo com o autómato desligado.

A capacidade de memória de cada autómato tem em conta as potencialidades de cada um e é geralmente medida em termos do número máximo de instruções de um programa, ou em termos da capacidade de memória em *bytes*.

A classificação dos autómatos programáveis é baseada no número de entradas e saídas envolvidas e no tipo de acções que podem executar, dividindo-se em duas grandes classesg

- Compactos
- Moldadores

Os de classe compactos são constituídos por um corpo único eventualmente expansível com módulos de entrada e saída lógica.

Os de classe moldadora são constituídos em módulos, cada um com o seu tipo de função. Estes módulos são facilmente ligados ou desligados conforme a aplicação corrente.

Os autómatos programáveis ainda podem se dividir em dois grupos

- Gama baixa
- Gama de média e alta

Os autómatos programáveis de gama baixa apenas possuem unidades lógicas de entrada e saída; o número de entradas/saídas disponíveis é baixo, sendo fixo ou expansível até 128 entradas/saídas e têm em geral uma construção compacta.

Os autómatos programáveis de gama média e alta têm uma construção modular; permitem a adição de módulos lógicos e digitais e podem ser ligados em rede. O que distingue os autómatos de gama média e alta é essencialmente a sua capacidade de processamento e a memória disponível.

Na automação programável o equipamento é projectado com a capacidade de mudar a sequência de operações de forma a acomodar diferentes configurações. A sequência de operações é controlada por um programa.

Os automatismos, sistemas que permitem a realização automática de operações, podem ser implementados através:

- **Tecnologia cablada**: o funcionamento do automatismo é determinado pela forma de ligação dos condutores (cablagem) entre os diferentes constituintes do automatismo (relés, temporizadores, etc.).
- **Tecnologia programável**: o elemento de comando é o autómato programável e o funcionamento do automatismo é determinado pelo programa armazenado na memória do autómato.

Na *tecnologia cablada*, qualquer modificação no funcionamento do automatismo implica modificar fisicamente a cablagem e, normalmente, novos componentes. Também é obrigatório parar o processo de fabrico, uma vez que é necessário realizar trabalho de desmontagem/montagem. O custo final é apreciável.

Na *tecnologia programável*, as modificações são efectuadas, mesmo quando o sistema estiver instalado, sem alteração da cablagem, de forma rápida e simples, uma vez que basta alterar ou trocar o programa do autómato. A flexibilidade é grande e o custo final é baixo.

A tecnologia cablada é utilizada em problemas simples. A tecnologia programável é mais utilizada em problemas de complexidade média, em pequenas linhas de produção, e alta (controlo de processos industriais), onde se exige flexibilidade e possibilidade de evolução.

O sistema de entradas/saídas fornece a ligação física entre o CPU e o processo a controlar. O autómato, através de sensores apropriados, pode medir quantidades físicas como velocidade, temperatura, pressão, corrente, etc. Baseando-se nos valores medidos, e no programa de controlo, o CPU controla as saídas que poderão actuar em dispositivos como, por exemplo, válvulas, motores, alarmes.

4.1. Material utilizado

O autómato escolhido para o desenvolvimento do trabalho é da Siemens séri S7-200. A tradução poderá ser feita para qualquer tipo de autómato, desde que seja conhecida a sua lista de instruções.

Resumindo, a Unidade Central de Processamento (CPU) é responsável pela execução do programa, controlando todas as operações dentro do autómato, através de instruções armazenadas na memória de programa. Um barramento de dados transporta a informação da memória e do sistema de entradas saídas para o CPU e vice-versa. Na maioria dos autómatos (principalmente os mais modernos) o CPU é baseado em um ou mais microprocessadores e outros circuitos que permitem realizar as funções de controlo e cálculo necessárias à execução de programas.

As principais caraqueteristicas que definem o CPU222 são:

- Uma fonte de carga de 24V e de 180mA, tambe pode funcionar como fonte de alimentação
- Contem quarto contadores rápiodos (30kHz). Cada contador provene 2 entradas separadas contendo um impulso de 20kHze com um deslocamento de 90°
- Impulso de saida máximo de 20kHz
- Contem uma memória máxia de124 ciclos por cada entrada dos temporizadores, contadores e memorias.

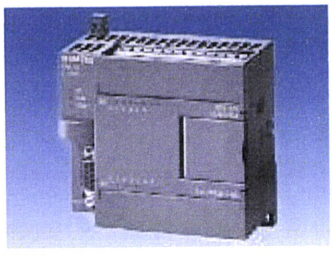


Fig. 7 – Imagem do CPU222

A característica que define o CP235 é ter quatro entradas e quatro saídas analógicas.

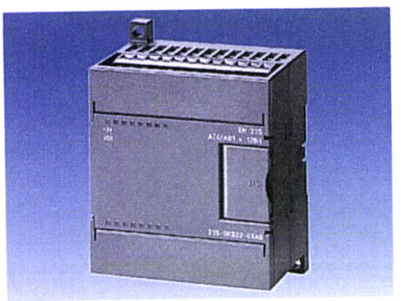


Fig. 8 – Imagem de CP235

4.2. Linguagem do autómato

Uma das programações utilizada é LADDER que tem uma parte integrante do software STEP 7. Isto significa que é instalada junto com o software STEP 7, toda a função do editor, do compilador, e teste (debug) para LADDER está disponível após a instalação do programa STEP 7. Há três linguagens de programação no software standard, STL, FBD, e LADDER.

STL é a linguagem escrita na forma de um editor de instruções, nos moldes de linguagem de alto nível como o "BASIC" ou a linguagem de programação "C".

FBD é uma linguagem gráfica em que o programador usa funções lógicas na forma de blocos, permitindo-lhe montar circuitos de forma semelhante a um circuito electrónico digital.

LADDER significa lógica de relê, é um dos processos mais usados na programação do CLP. Este também é um tipo de linguagem gráfica, em que o técnico programador projecta o programa como se estivesse desenhando um circuito eléctrico.

Este programa converter um programa de uma linguagem para outra quase sem restrição, e escolhe o idioma mais apropriado para um bloco particular que esteja programando. Se o programa for escrito em LADDER ou FBD, ele sempre pode ser trocado para a Representação STL. Se o programa for escrito em LADDER podem ser convertidos em FBD e vice-versa, no entanto nem sempre esta conversão pode ser feito, elementos de programa que não podem ser representados na linguagem de destino são exibidos em STL. A entrada das variáveis do bloco de dados, que trabalham bancos de dados, pode ser feita de forma simples com ajuda de janelas de edição.

Instruções LADDER consistem em elementos e caixas que estão conectados graficamente formando redes (Network) de um circuito eléctrico. As funções lógicas são tratadas como circuitos eléctricos contendo elementos do tipo contactos e caixas contendo circuitos especiais, estas caixas possuem conexões de entrada e saída.

Um endereço é a descrição na programação LADDER que relaciona a instrução lógica, na forma de contacto ou caixa, com um endereço da memória interna do CLP. Esta memória estar relacionada com a entrada ou de saída do CLP. Do ponto de vista do

endereço existem instruções de entrada e de saída, a instrução de entrada lê o dado da memória e a instrução de saída escreve o dado na memória.

O CLP possui memória interna, como um computador, é registado e armazenado os dados de operações lógicas e matemáticas. Existe um ficheiro chamado de status que armazena uma palavra onde os bits dão informações sobre como a instrução se desenrolou. No ficheiro Status existe um bit muito importante chamado de RLO (Resultado Lógico da Operação) que armazena o resultado lógico do fluxo lógico ao longo do circuito da network.

Como num circuito lógico o que percorre o circuito é a corrente eléctrica, na lógica LADDER o que percorre o circuito é o Fluxo de Energia Lógico.

O fluxo de energia percorre o circuito da esquerda para a direita e de cima para baixo, o programador deve imaginar que a fonte de alimentação do circuito está a esquerda da network.

A metodologia de tradução proposta considerará que o programa de controle PLC é escrito como uma linguagem de programação orientada por texto, em um arquivo de texto padrão. Isto não representa um problema porque cada PLC tem um ficheiro que programa software permitindo salvar programa PLC num ficheiro nesse formato. Os PLC controlam o programa neste formato. A fig. 9. Mostra um exemplo de arquivo de texto de Programa de Controle de PLC simples, como um exemplo, um Siemens PLC.

```
1   //
2   // PROGRAM TITLE COMMENTS
3   //
4   NETWORK 1
5   LD    I 0.0
6   A     I 0.1
7   LD    I 0.2
8   A     I 0.3
9   OLD
10  =     Q 0.0
11  //
12  NETWORK 2
13  LD    I 0.4
14  LD    I 0.5
15  CTU   C5, +6
16  //
17  END
```

Fig. 9 - PLC Control Program standard text file.

O programa de controlo PLC, a ferramenta de tradução é um software, desenvolvido em Visual C + +, o que converte automaticamente o programa PLC controle em um arquivo de texto correspondente Matalb / Simulink m-arquivo. Este m-arquivo, contendo o programa descrito no controle PLC Matalb / Simulink língua, detém uma função Matlab / Simulink. Conhecendo o PLC do número de entradas / saídas, a ferramenta de conversão fixa o número correcto de argumentos de entrada e saída da função. A tradução do próprio Programa PLC Controle relés sobre um conjunto de regras de tradução para o conjunto de instruções PLC lista.

Uma lista completa PLC instrução pode ser dividido em:

- Boolean
- Comparação
- Saída
- Timer
- Contador
- Matemática
- Incremento / Decremento
- Mudança / Transferir
- Programa Controle
- Outros

5. Matlab/Simulink

Em variados meios industriais e académicos utiliza-se o MATLAB por constituir um software interactivo de alta performance direccionado para o cálculo numérico. O MATLAB permite a realização de aplicações ao nível da análise numérica, de análise de dados, cálculo matricial, processamento de sinais e construção de gráficos, entre outras, já referidas.

O MATLAB é um sistema interactivo cujo elemento básico de informação é uma matriz que não requer dimensionamento.

O simulink é uma ferramenta de simulação integrada no software do Matlab baseada em diagramas de blocos que permite modelizar e analisar sistemas dinâmicos. Permite construir diagrama de blocos, simular sistemas não lineares, permite fazer simulação contínua ou discreta e integração com o Matlab.

Como referido antes, que a metodologia de tradução proposta assuma que o processo industrial já é simulado no ambiente Matlab/Simulink, como apresentado na Figura.10.

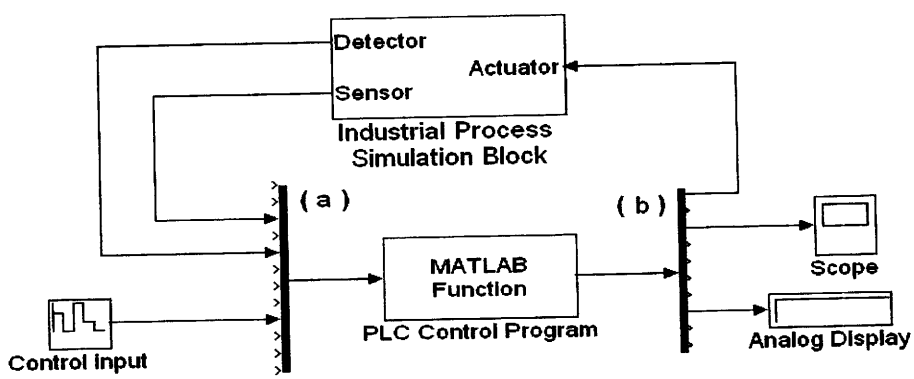


Fig. 10 – Operação de PLC e processo industrial interactiva.

O processo industrial controlado PLC é simulado num bloco de Matlab/Simulink denominado 'Bloco de Simulação de Processo Industrial'. Este bloco contem os sensores de processo e sinais de booleanos, que serão usados como entradas ao bloco de Matlab/Simulink denominado 'Programa de Controle de PLC'. Este bloco emulará a operação PLC e as suas produções corresponderão às funções PLC que se unirão à entrada de accionadores, em 'o Bloco de Simulação de Processo Industrial'.

O bloco 'Programa de Controle de PLC' é a chave da metodologia proposta. Ele emulará a operação PLC do modo cíclico que foi apresentado no Figo 6. Este bloco de função é descrito num ficheiro de extensão m.

O relógio de onda centrada colocado nas entradas do PLC é para o ciclo do PLC ocorra numa forma discreta. Na ausência deste o PLC não funcionará correctamente. O relógio regula os ciclos de x a x tempo, quando o sinal de entrada for 1 o PLC percorre um ciclo completo.

5.1. Modelagem do processo industrial

A implementação de uma linha de produção industrial implica um grande investimento. Cada decisão na etapa de desenho deve ser tomada com extremamente cuidado para assegurar que o processo de fabricação automatizado realizará com sucesso os resultados desejados.

O problema da modelagem de processo industrial não é fácil ou único, e várias aproximações podem ser tomadas. Esses processos podem ser considerados como um sistema de evento discreto, onde os estados do sistema dinâmico implicaram modificações consequentemente às ocorrências de eventos vários e distintos. É importante, guardar as manufacturas e processar a competitividade de integradores, que os sistemas de produção industriais continuam ser constantemente melhorados. Para realizar este objectivo uma aproximação de modelagem eficiente é uma questão fundamental. Uma linha de produção moderna é um sistema altamente integrado composto de estações de trabalho automatizadas como robôs com capacidades modificam instrumento, um sistema de manejo de hardware e sistema de armazenamento, processos de controlou PLC, e um sistema de controle de computador que controla as operações do sistema inteiro. Cada processo industrial controlado pode ser modelado como uma função de transferência (contínuo e/ou discreto) com um jogo de entradas e um jogo de produções, como apresentado no Figo 11. As entradas referem-se aos sinais de controle que são aplicados aos accionadores de processo, e as produções referem-se às variáveis que são adquiridas pela rede de processo de sensores e detectores.

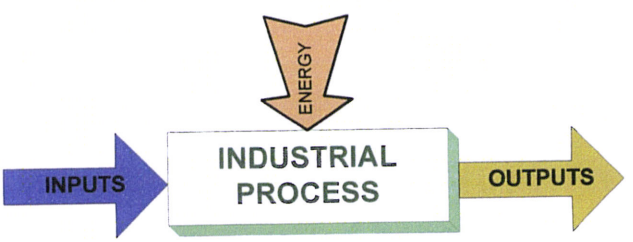


Fig. 11 – Modelo do processo industrial.

Embora Matlab/Simulink muitas vezes não seja tão usado na modelagem de processos industrial, este bloco de simulação permite uma modelagem cómoda e o instrumento de simulação para contínuo, discreto e variado discreto/contínuo modelos de subsistemas. Actualmente, Matlab/Simulink cobre uma larga variedade de áreas de aplicação e também pode ser usado para construir modelos de processo detalhados em aplicações de automação.

O PLC que modela questão pode ser reduzido à emulação do programa de controle de PLC. Várias aproximações podem ser tomadas quanto ao programa PLC. Vários autores desenvolveram listas específicas da verificação do programa PLC. Essas listas verificam a estrutura da utilização de programa, entre outros, redes de autómatos. Muitas vezes esses programas só verificam a estrutura de programa sem verificar se ele realizar os objectivos de controle desejados. Outra aproximação é a geração do programa PLC de outro formalismo, como redes petri, diagramas estatais ou máquinas estatais finitas. Se o formalismo original é o erro gratuito isto pode ser um instrumento valioso para desenvolver programas PLC. Alguns autores desenvolveram lista de software para traduzir programas PLC ao código de DSP, para que possa ser usado no hardware non-PLC.

Nenhuma dessas aproximações é destinada para ser usado dentro do ambiente Matlab/Simulink. A aproximação de metodologia proposta considerará que o PLC é essencialmente modelado emulando o seu programa de controle, que interage com o próprio processo industrial controlado, como apresentado no Figo 12.

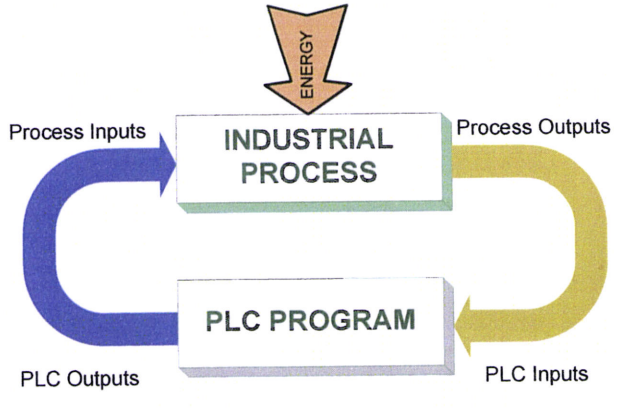


Fig. 12 – Programa de controlo e processo interactivo industrial.

5.2. Instrução de Função

A escolha do tipo PLC é essencial para o estabelecimento das regras de tradução de acordo com o fabricante. Embora todos eles são baseados lógica booleana, cada fabricante de PLC desenvolve sua própria programação de sintaxe. Desta forma, deve-se saber a lista de instruções do PLC dado pelo fabricante, a fim de aplicar as normas adequadas de tradução.

O número de Entradas de PLC é claramente os argumentos da função de Matlab/Simulink 'Programa de Controle de PLC'. Esta função será responsável por executar o programa de controle de PLC dentro do ambiente Matlab/Simulink, e será criada como um ficheiro de texto .m. Os di1 representam as entradas digitais do PLC, ai1 as entradas análogas do PLC, do1 a dop são as saídas digitais do PLC e ao1 a aoq são as análogas do PLC. os n, o m, p e q indicam, respectivamente, o número de entradas digitais, entradas análogas, saídas digitais e análogas do PLC.

É importante notar que n + m define a dimensão do bloco MUX (a) na fig. 6. Do mesmo modo, p + q definir a dimensão do bloco Demux (b) na fig. 6.

$$
\begin{array}{l}
function \quad [output\ ] = \\
= \text{PLC Control Program } (di_1, \ldots, di_n, \ldots, ai_1, \ldots, ai_m) \\
\ldots \\
\left(\begin{array}{l} \text{PLC Control Program} \\ \text{in Matlab/Simulink language} \end{array} \right) \\
\ldots \\
output = [do_1, \ldots, do_p, \ldots, ao_1, \ldots, ao_q]
\end{array}
\qquad (1)
$$

5.3. Instruções lógicas

Em qualquer sistema digital a unidade básica construtiva do modelo dominante são as portas lógicas. As portas lógicas podem se encontrar ao nível de integração de larga escala, por exemplo, nos circuitos integrados de processadores Pentium. Ou então a nível de integração existente em circuitos integradores digitais mais simples.

Estas têm facilidade de processamento de números binários decorre da existência de apenas dois dígitos, 0 e 1 (bit), que podem ser representados por 2 níveis de tensão (por exemplo 0 = 0 volt e 1 = 5 volts).

As instruções Booleanas serão traduzidas em Matlab / Simulink utilizando linguagem padrão Matlab das funções booleanas, tal como apresentado no Quadro I, onde xy é uma entrada digital e Q xy é uma saída digital, x e y são, respectivamente, os bits e bytes são considerados digitais nas entradas / saídas. Além disso, do_g é uma variável Matlab sendo a saída digital g e di_h é uma variável Matlab designada por entrada digital h. O VERDADEIRO estado booleano será representado em ambiente Matlab por '1 ' e FALSO por '0'.

Instruções Boolean	Instruções PLC	Matlab/Simulink
AND	LD I a.b A I c.d = Q e.f	do_g = di_h & di_i
OR	LD I a.b O I c.d = Q e.f	do_g = di_h | di_i
NOT	LDN I a.b = Q c.d	do_g = ~ di_i

Quadro I – Tradução das portas lógicas

Combinações de várias instruções Booleanas serão convertidas utilizando as regras mencionada no quadro I. Como um exemplo, o conjunto de instruções (2), serão representados como (3).

```
LD   I 0.0
A    I 0.1
LD   I 0.2                    (2)
OLD
=    Q 0.0
```

do_1 = (di_1 & di_2) | di_3 (3)

5.4. Temporizadores e contadores

Os temporizadores e contadores são geralmente utilizadas para activar ou desactivar um dispositivo ao fim de determinado tempo ou contagem.

O seu princípio de funcionamento é idêntico pois ambos podem ser considerados contadores. Um temporizador conta um número de intervalos de tempo fixos, necessário para atingir a duração pretendida, enquanto o contador regista o número de ocorrências de um determinado evento.

As instruções de temporização e contagem necessitam de dois registos: um registo para armazenar o número de contagens já efectuadas e outro registo para armazenar o valor inicial.

5.4.1. Contador

O contador como o nome indica é o número de vezes que ocorre um determinado acto. Existe dois tipos de contadores:

- Count Up;
- Count Up/Down.

Em LAD são representados como demonstra a fig.9, respectivamente

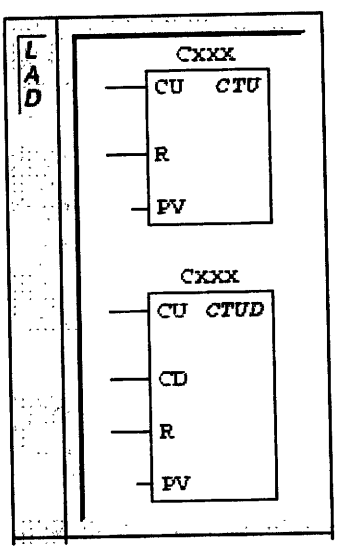


Fig.13 – Ilustração dos dois tipos de contadores em LAD

Simbologia dos inputs:

CU - Contador crescente

CD – Contador decrescente

R – Reset

CTU – Count Up

CTUD – Count Up/Down

PV - Preset Value (número de repetições que o contador tem que contar)

A instrução de CTU incrementa de uma unidade o seu valor acumulado de cada vez que ocorre um evento de contagem. O contador, na maioria das implementações, continua a contar eventos mesmo após o seu valor de contagem ser atingido.

O princípio de funcionamento do contador decrescente é idêntico ao contador ascendente, só que o processo de contagem é desde o valor pré-programado até zero. Também é normal haver outras implementações de contadores nomeadamente o contador ascendente/descendente, bem como a introdução de uma entrada para inicialização (*reset*) em cada um dos contadores atrás citados. Quando a entrada de *reset* está inactiva o contador tem um comportamento normal, e uma activação da entrada de *reset* inibe imediatamente a contagem e inicializa o contador com o seu valor pré-programado.

No diagrama da fig.10 está representado o funcionamento dos dois contadores com os respectivos impulsos e reset. No inicio a entrada I4.0 o contador crescente começa no zero. Algum tempo depois entra o sinal 1 onde começa a contar voltando ao zero sempre assim sucessivamente, por cada vez que ele volta a 1 ele avança uma unidade na contagem. Quando o contador crescente (count up) pára, o sinal mantém-se a zero. O contador decrescente nesta altura passa de zero a 1 fazendo com que a contagem diminua, funcionando do mesmo modo que o contador crescente. O Reset faz com que os contadores fiquem a zeros na sua contagem e isso acontece quando este passa de zero para 1.

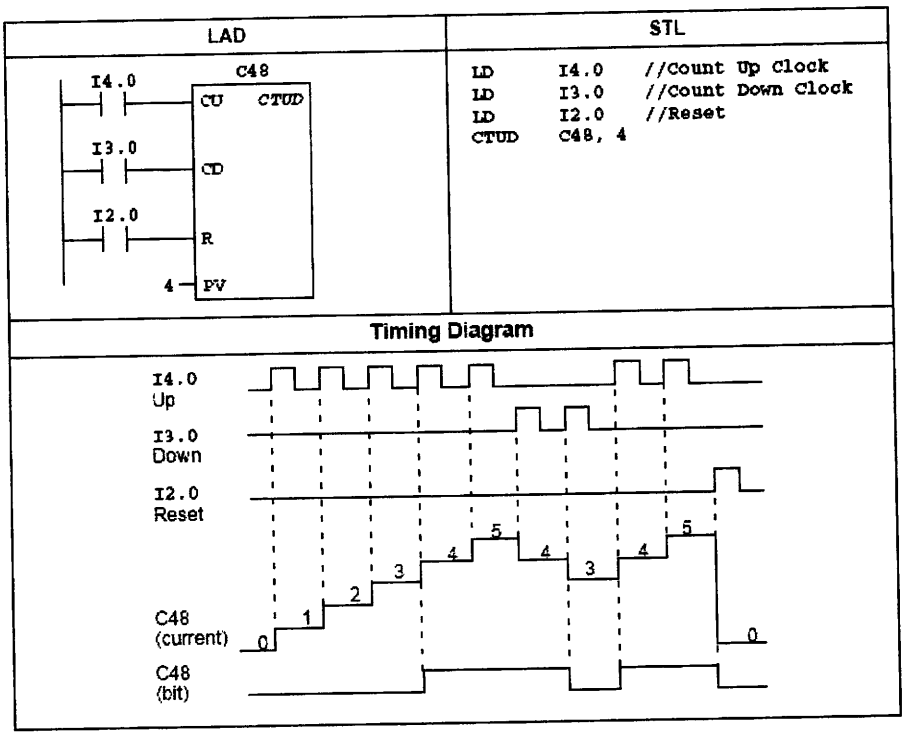


Fig.14 - Exemplo de instruções de um contador em linguagem LAD e STL. (Siemens, Simatic S7-200 Programmable Controller, System Manual)

A instrução contador (CTUD - contador ascendente e descendente) pode exigir várias entradas booleanas: uma contagem crescente, outra para a contagem decrescente (se for o caso) e outras para reset do contador. Uma vez que a contagem só é realizada na passagem de zero para um da entrada booleanas, a tradução Matlab / Simulink deve ter em conta o estado anterior nas entradas booleanas. Nas expressões a (4) e (5) apresenta um exemplo de contagem, onde di_1_prev é uma variável Matlabb denotando o estado anterior da variável di_1. Este estado indica o estado anterior do ciclodo programa de controlo do PLC. Neste exemplo, ao atingir o valor 4, o contador muda para o estado verdadeiro. No ambiente Matlab c_10 designa o estado do contador booleano número 10, e contador c_10_value indica o mesmo valor do contador.

```
LD    I 0.0  // Count up
LD    I 0.1  // Count down
LD    I 0.2  // Reset counter
CTUD  C10, +4
```
(4)

```
if di_1 & ~ di_1_prev
    c_10_value = c_10_value + 1
end
if di_2 & ~ di_2
    c_10_value = c_10_value - 1
end
if di_3
    c_10_value = 0
    c_10 = 0
end
if c_10_value >= +4
    c_10 = 1 end
```

(5)

5.4.2. Temporizador

O TEMPORIZADOR é semelhante a um cronómetro, conta o tempo (s). Existe dois tipos de temporizadores:

- On-Delay Timer;
- Retentive On-Delay Timer.

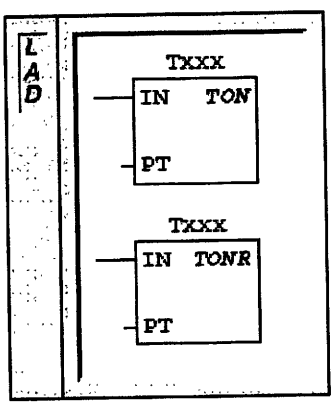


Fig. 15 – Imagem de dois tipos de temporizador em LAD

Simbologia:

IN – entrada do temporizador

PT - Preset Time (tempo que o temporizador está a funcionar)

TON - On-Delay Timer

A temporização é iniciada quando a condição de entrada é verdadeira, sendo o temporizador inicializado quando a condição de entrada deixar de ser verdadeira.

Quando a lógica do *rung está* ligado à entrada de controle do temporizador e tiver continuidade lógica (lógica ON), o temporizador começa a contagem dos *ticks*. Durante o processo de contagem, o valor do registado acumulador é incrementado, a cada unidade de tempo da base de tempo, desde 0 até ao valor contido no registador de *preset* (Dependendo do CLP), o processo pode ser invertido. O registador acumulador recebe o valor do registador de *preset* no início do processo. A cada *tick*, o registador acumulador é decrementado e quando ele atingir 0 a saída é accionada. Quando estes

dois valores forem iguais, o temporizador activa a saída. Quando a continuidade lógica da entrada de controle for perdida (lógica OFF), o processo de contagem de *ticks* é parado, o valor do registador acumulador é zero e a saída do temporizador é desligada.

O diagrama a baixo demonstra o funcionamento do temporizador TNOR. Ele só é activado quando na sua entrada I2.1 tem um sinal de 1 e ao fim de 10 segundos este recebe um sinal zero, parando assim a sua contagem e ao fim de um determinado tempo é activado e continua a contar o tempo do "sítio" de onde parou.

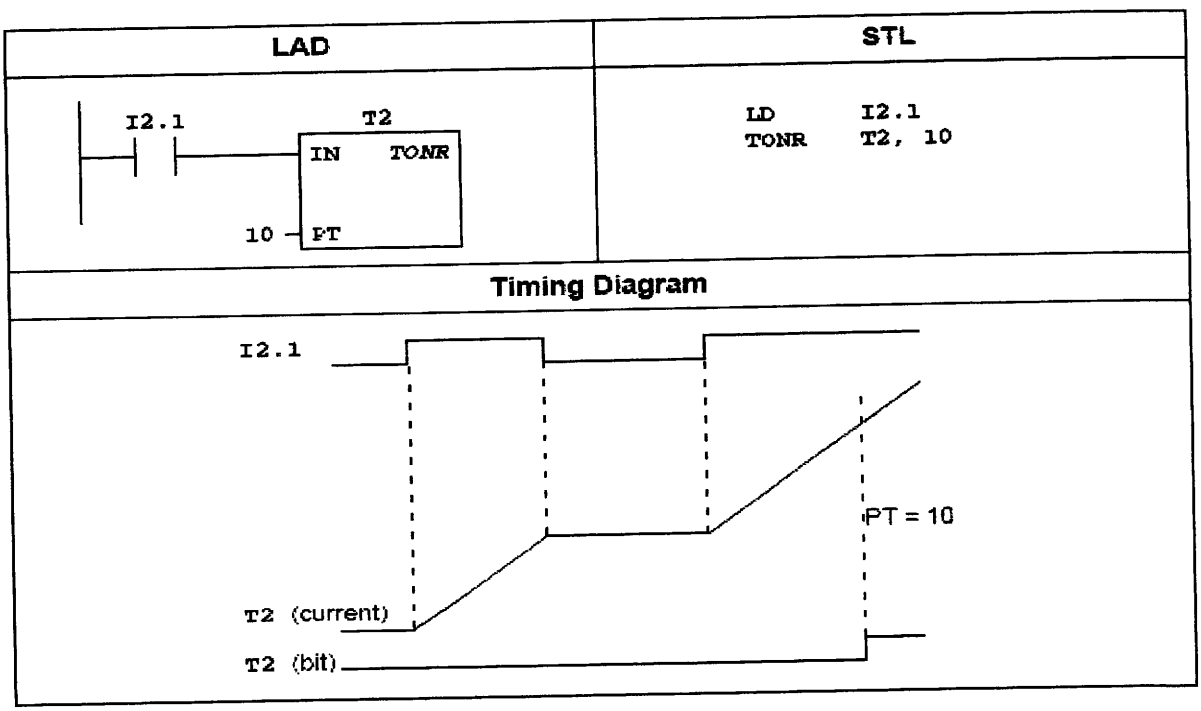


Fig. 16 - Exemplo de instruções de um contador em linguagem LAD e STL. (Siemens, Simatic S7-200 Programmable Controller, System Manual)

Para representar o temporizadro em ambiente Matlab é necessário ir buscar o temporizador do comptador (second = second (now)), para que este faça uma contagem mais exacta. Como contador é preciso indicar o estado anterior, para se manter sempre actalizado. O estado anterios neste caso está representado por di1_b, qando o tempo final for menor que o o tempo do PC o temporizador desliga

```
if di1==1 & di1_b==0
    t11=0;
    temp_inic=second (now);
    temp_final=temp_inic+1;
end
if second (now)>temp_final
    t11=1;
end
```

5.5. Instruções aritméticas

Estas instruções incluem as quatro operações aritméticas básicas: adição, subtracção, multiplicação e divisão. Existem, no entanto, autómatos que fornecem outro tipo de operações aritméticas como, por exemplo, a raiz quadrada, complemento, incremento/decremento.

Operações disponíveis no Matlab:

+ Adição
- Subtracção
* Multiplicação
\ Divisão à esquerda
/ Divisão à direita

Tais operações aplicam-se também a escalares (considerados matrizes de 1 x 1).

Se os tamanhos das matrizes são incompatíveis, para a operação matricial, será gerada uma mensagem de erro, com excepção do caso de operações entre escalares e matrizes (para adição, subtracção, divisão e multiplicação), quando cada entrada da matriz é operado pelo escalar.

5.5.1. Adição

Adição é uma das operações básicas da álgebra. Na sua forma mais simples, adição combina dois números (termos ou parcelas), em um único número, a soma. Ao adicionar mais números corresponde a repetir a operação. Se os termos são escritos individualmente, então a adição é escrita usando-se o sinal mais ("+"). Assim, a soma de 1, 2 e 4 é escrita como 1 + 2 + 4 = 7.

Instruções aritméticas dos PLC's são geralmente booleanas activas. Isto implica a utilização em Matlab a função 'se', a fim de representar o seu comportamento. Como um exemplo, a instrução de adição de PLC (6) é representada como (7) no ambiente Matlab / Simulink. Onde AIW0 e AQW0 representam, respectivamente, num PLC entrada analógica e saída analógica deste. ao_1 em Matlab é uma variável que representa a primeira saída analógica e ai_1 Matlab é uma variável que simboliza a primeira entrada analógica.

```
LD    I 0.0
+I    AIW0 , AQW0                    (6)
```

```
if di_1
   ao_1 = ai_1 + ao_1                (7)
end
```

5.5.2. Subtração

Subtracção é uma operação matemática que indica quanto é um valor numérico (*minuendo*) se dele for removido outro valor numérico (*subtraendo*). Uma subtracção é representada por:

$$a - b = c$$

a é o minuendo, b é o subtraendo, c é a diferença.

A subtracção é o mesmo que a adição por um número de sinal inverso. É, portanto, a operação inversa da adição. A instrução aritmética é análoga à soma.

5.5.3. Multiplicação

Em matemática, a multiplicação é uma operação binária. Na sua forma mais simples a multiplicação é uma forma simples de se adicionar uma quantidade finita de números iguais. O resultado da multiplicação de dois números é chamado *produto*. Os números sendo multiplicados são chamados de coeficientes ou operandos, e individualmente de multiplicando e multiplicador.

A multiplicação pode ser escrita de várias formas equivalentes. Todas as formas abaixo significam, "5 vezes 2":

5×2

$5 \cdot 2$

$(5)2,\ 5(2),\ (5)(2),\ 5[2],\ [5]2,\ [5][2]$

$5 * 2$

O asterisco é usado frequentemente em computação pois em um símbolo existente em todos os tipos de teclado, mas não é usado quando escrevendo-se matemática à mão (A origem desta notação vem da linguagem de programação FORTRAN).

Na instrução aritmética em linguagem PLC representa-se

```
LD      I0.0

MOVW        +6, VW6

MUL  +9, VD4
```

Onde +6, +9 e VW6 são entradas analógicas do autómato, VD4 será ma saída analógica.

Na linguagem do Matlab é necessária uma condição if , onde ao_1 é a primeira saída analógica do autómato, ai_1 e ai_2 é a primeira e segunda, respectivamente, entradas do autómato analógicas.

```
If di_1

ao_1=ai_1*ai_2

end
```

5.5.4. Divisão

Divisão é a operação matemática que determina a quantidade de vezes que um número (*divisor*) está contido dentro de outro número (*dividendo*). A divisão é a operação inversa da multiplicação.

De um ponto de vista informal, divisão é o acto de dividir, repartir, separar as partes de um todo. O resultado de uma divisão de um número não-nulo por 0 tende ao infinito. O resultado de 0 dividido por 0 é indeterminado; o resto de uma divisão nunca pode ser um número maior ou igual ao divisor.

Existem algumas formas de se representar uma divisão:

- Como uma fracção: $\frac{a}{b}$
- Com uma barra: a/b
- Com o sinal de divisão: $a \div b$
- Usando dois pontos: $a : b$
- Usando o sinal de inverso: ab^{-1}

a é o "dividendo" e b é o *divisor*. O resultado de uma divisão é denominado *quociente*.

Os termos da divisão são = dividendo, divisor, resto e quociente.

Em ambiente de trabalho de PLC a divisão é semelhante à multiplicação.

Em ambiente de Matlab

```
If di_1

ao_1 = ai_1 / ai_2

end
```

5.6. Incremento/decremento

O incremento/decremento unitário de uma variável é uma operação tão frequente em programação o que existem operadores específicos para o efeito

- Operadores de incremento: ++.
- Operadores de decremento: --.

Exemplos:

- **x++;** (equivalente a **x+=1;**)
- **y--;** (equivalente a **y-=1;**)

Os operadores de incremento/decremento podem ser utilizados como *prefixo* ou *sufixo* do operando e aplicados no contexto de uma expressão mais complexa.

– Quando utilizados como prefixo, o valor da variável é incrementado (ou decrementado) antes de ser utilizado no contexto da expressão em que se encontra.

– Quando utilizados como sufixo, o valor da variável é incrementado (ou decrementado) depois de ser utilizada no contexto da expressão em que se encontra.

Em ambiente de Matlab

```
If di_1
ao_1 = ai_1+1
End
```

5.7. Instruções de comparação

Os operadores de comparação podem ser aplicados a todos os tipos de variáveis elementares (números, detalhes de data, sequências e valores booleanos).

= Igualdade de números, valores de data e sequências

<> Desigualdade de números, valores de data e sequências

> Maior do que verifica números, valores de data e sequências

>= Maior do que ou igual a para verificar números, valores de data e sequências

<Menor do que verifica números, valores de data e sequências

<= Menor do que ou igual a para verificar números, valores de data e sequências

A representação em LADDER

LDB>= 5, 4

= Q0.6

Em Matlab representa-se

If di_1 = =1

4<ai1=<5

end

5.8. Instruções de deslocamento

Um registo de deslocamento na linguagem LAD necessita de duas etiquetas que identificam o endereço das palavras inicial e final.

Exemplos de nomes para identificação de alguns símbolos:

LN representa uma ligação horizontal

IO representa um Contacto Normalmente Aberto

IC representa um Contacto Normalmente Fechado

OO representa uma Saída Normalmente Aberta

UDC representa um Contador Ascendente/Descendente

...

SFT representa um Registo de Deslocamento

5.8.1. Grafcet

O grafcet é um método gráfico de apoio à concepção de sistemas industriais automatizados, que permite representar, através de modelos do tipo dos gráficos de estados, o comportamento de sistemas sequenciais. A partir do modelo Grafcet de um sistema, pode ser gerado automaticamente o programa do controlador desse sistema, sendo muito mais simples construir o modelo Grafcet, do que desenvolver o programa do controlador.

O GRAGCET possibilita descrever o funcionamento de sistema complexos através de modelos compactos e, dessa forma, estruturara concepção desses sistemas; simular o funcionamento dos sistemas e, assim, detectar e eliminar eventuais erros de concepção antes de passar à fase de implementação.

Relativamente aos gráficos de estados convencionais, o Grafcet introduz a possibilidade de representar a sincronização de operações entre subsistemas, que podem decorrer em simultâneo ou em alternativa; o funcionamento dos sistemas segundo vários níveis de detalhe e as interacções entre sistemas de controlo organizados hierarquicamente.

Na concepção dos sistemas industriais automatizados, a construção do modelo Grafect é uma etapa intermédia entre a especificação do sistema e o programa final.

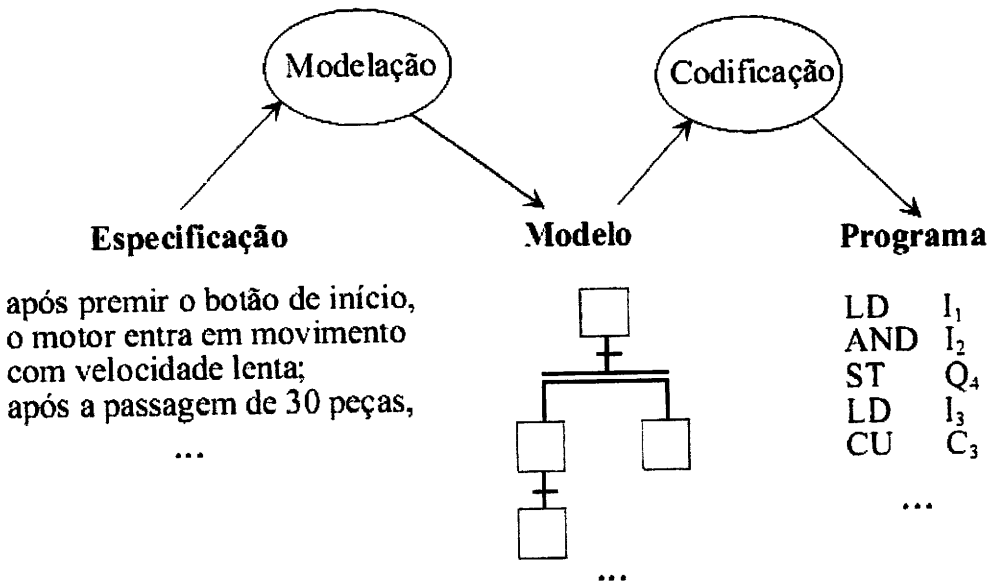


Fig. 17 – Esquema da interacção entre o programa, grafcet e as especificações do processo industrial.

Num modelo Grafcet, tipicamente:

- Os sinais provenientes dos detectores estão associados às receptividades
- As ordens enviadas aos actuadores estão associadas às acções
- As etapas e as transições definem a estrutura do programa do controlador

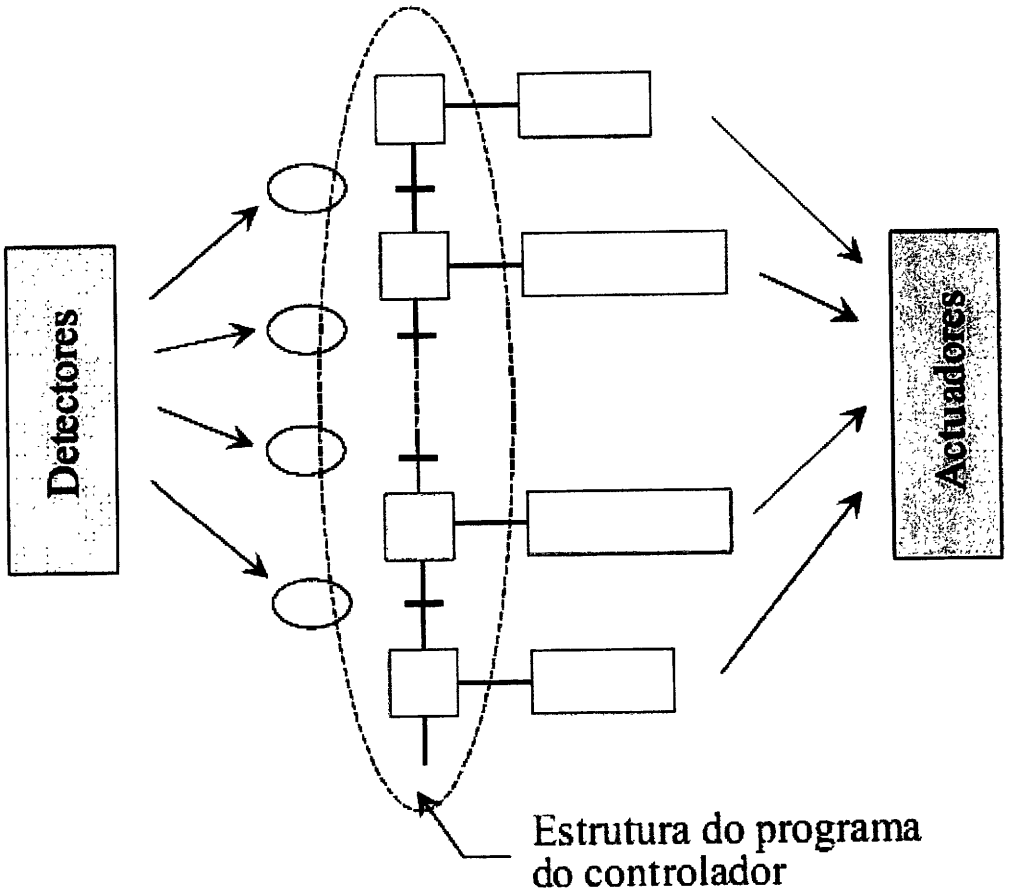
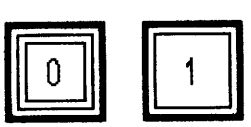


Fig. 18 – Esquema do Grafcet.

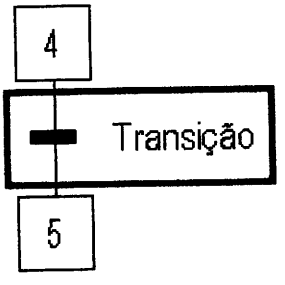
Uma etapa é definida como a situação do sistema que representa um estado invariante no que diz respeito às entradas e saídas do automatismo. Num determinado momento e de acordo com a evolução do sistema, uma etapa pode estar activa ou inactiva. Diz-se que a etapa está activa, quando são executadas sobre o processo as tarefas elementares programadas.

Graficamente é simbolizada por um rectângulo com uma numeração no seu interior, dando desta maneira uma sequencialidade das etapas representadas. A entrada e a saída de uma etapa aparecem na parte superior e inferior, respectivamente, de cada símbolo.



Nas etapas e só nelas são realizadas as acções e eventualmente pode não se realizar qualquer acção. Em cada instante, numa dada sequencia só uma etapa é que está activa.

O conceito de transição esta associado à barreira existente entre duas etapas consecutivas e cuja activação torna possível a evolução lógica do automatismo. A toda a transição é correspondida uma receptividade, que é a condição lógica necessária para que se execute uma acção da etapa seguinte, se bem que esta será executada sempre que a etapa precedente está activa. A condição lógica vem expressa mediante uma função lógica booleana. A sua representação gráfica consiste numa barra perpendicular à linha orientada associada.



6. Tradutor

Para fazer a tradução é necessário interligar com as duas linguagens, Matlab e LAD. O programa Matlab só lê ficheiros do tipo m e LAD expressa-se em txt, convertendo assim de m ficheiro txt para m, para tal utilizou-se o programa Visual Basic C++.

Para compreender o funcionamento das instruções do autómato em LAD, construí-se um programa com as principais instruções no SIMATIC 7. Depois de fazer as equivalências entre as duas linguagens preparo-se para começar a tradução.

Para que o programa leia o ficheiro txt abre-se uma nova instancia com o nome do programa e a respectiva extensão. Este lê o ficheiro todo. O objectivo é ler linha a linha, para tal, é necessário colocar os códigos de comando do ficheiro txt em linha, para que se possa lê-lo a linha a linha. As network e comentários serão eliminados. Para tornar as definições de código mais claras separa-se as palavras. Feito isto está pronto para programar em Visual Basic, isto é, para traduzir o ficheiro txt. O programa lê o ficheiro e ao encontrar um determinado código é descodificado e será gravado no ficheiro (.m), que será lida depois pelo Matlab.

Em lista de comandos o contador up/douwn

LD I0.0

LD I0.2

LDN I0.7

CTUD C48, VW0

Onde a primeira linha é entrada CU, a segunda é CD, a terceira é o reset do contador, o CTUD é o tipo de contador, o C48 é o contador e VW 0 é onde são guardados os valores que correspondem a uma saída do autómato.

```
    'CASO DO CONTADOR
ElseIf linhas_novas(k, 1) = "CTUD" Then
      linhas_novas(k, 2) = "Cv"

      linha_direita(w) = "If " & linhas_novas(k - 3, 2) & " =1" & " &
" & linhas_novas(k - 2, 2) & "=0"
      w = w + 1
      linha_direita(w) = linhas_novas(k, 2) & "=" & linhas_novas(k, 2)
& "+1;"
      w = w + 1
      linha_direita(w) = "End"
       w = w + 1

      linha_direita(w) = "If " & linhas_novas(k - 3, 2) & " =0" & " &
" & linhas_novas(k - 2, 2) & "=1"
      w = w + 1
      linha_direita(w) = linhas_novas(k, 2) & "=" & linhas_novas(k, 2)
& "-1;"
       w = w + 1
      linha_direita(w) = "End"
      w = w + 1

      linha_direita(w) = "If " & linhas_novas(k - 3, 2) & " =0" & " &
" & linhas_novas(k - 2, 2) & "=0"
      w = w + 1
      linha_direita(w) = linhas_novas(k, 2) & "=0;"
      w = w + 1
      linha_direita(w) = "End"
       w = w + 1

      linha_direita(w) = "If Cv >= " & linhas_novas(k, 3)
      w = w + 1
      linha_direita(w) = "Cv=1;"
      w = w + 1
      linha_direita(w) = "End"
      w = w + 1

      linha_direita(w) = "If Cv < " & linhas_novas(k, 3)
      w = w + 1
      linha_direita(w) = "Cv=0;"
      w = w + 1
      linha_direita(w) = "End"
```

A representação do TON sistema de listagem

LD I0.0

TON T32, +5

Em que na primeira linha é dado o sinal para que se lige ou desligar o temporizador, o TON é o tipo e temporizador, T32 é o relógio e na treceira culuna é o tempo que o temporizador está ligado. O temporizador e ligado ao relogio do computador, "Seconds = second(now)", que conta o tempo em segndos.

```
        'CASO DO TEMPORIZADOR
ElseIf linhas_novas(k, 1) = "TON" Then
       linhas_novas(k, 2) = "Seconds = second(now)"

       linha_direita(w) = "If " & linhas_novas(k - 1, 2) & " =0"
       w = w + 1
       linha_direita(w) = linhas_novas(k, 2) & " =0"
       w = w + 1
       linha_direita(w) = "End"
       w = w + 1


       linha_direita(w) = "If " & linhas_novas(k, 2) & " =0"
        w = w + 1
       linha_direita(w) = "t11=0;"
        w = w + 1
       linha_direita(w) = "T33b=0"
        w = w + 1
       linha_direita(w) = "End"
        w = w + 1

       linha_direita(w) = "If " & linhas_novas(k - 1, 2) & " =1 "
       w = w + 1
       linha_direita(w) = linhas_novas(k, 2) & " = " & linhas_novas(k,
2) & "+1"
       w = w + 1
       linha_direita(w) = "End"
       w = w + 1

       linha_direita(w) = "if " & linhas_novas(k, 2) & ">Tin+3 "
       w = w + 1
       linha_direita(w) = "T33b=1"
       w = w + 1
       linha_direita(w) = "End"
        LD = w + 1
```

7. Aplicação

Um exemplo da execução do autómato é um controlo de nível de um tanque de água, cujo caudal de entrada de água no depósito é aleatório e varia entre 2 e 8m/s. O objectivo é manter o nível de água no depósito entre 2 e 5 metros de altura. As duas entradas são detectores de nível. A saída digital faz actuar uma válvula de saída de água que debita um caudal de 8m/s.

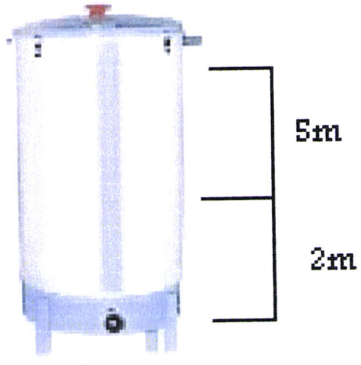


Fig. 19 – Figura do tanque de água

Como objectivo secundário queremos saber quantas vezes foi atingido o nível máximo de 5 metro, utilizando um contador.

Na figura que se segue é simulado, no Simulink, o depósito de água com a respectiva funcionalidade (contador). O depósito é representado pelo bloco cujo sinal é aleatório. O rectângulo azul é o bloco que ocorre o controlo do nível de água no depósito. As saídas (nível máximo e mínimo do deposito) deste bloco interagem com as entradas do bloco Matlab Function onde se encontra o programa.

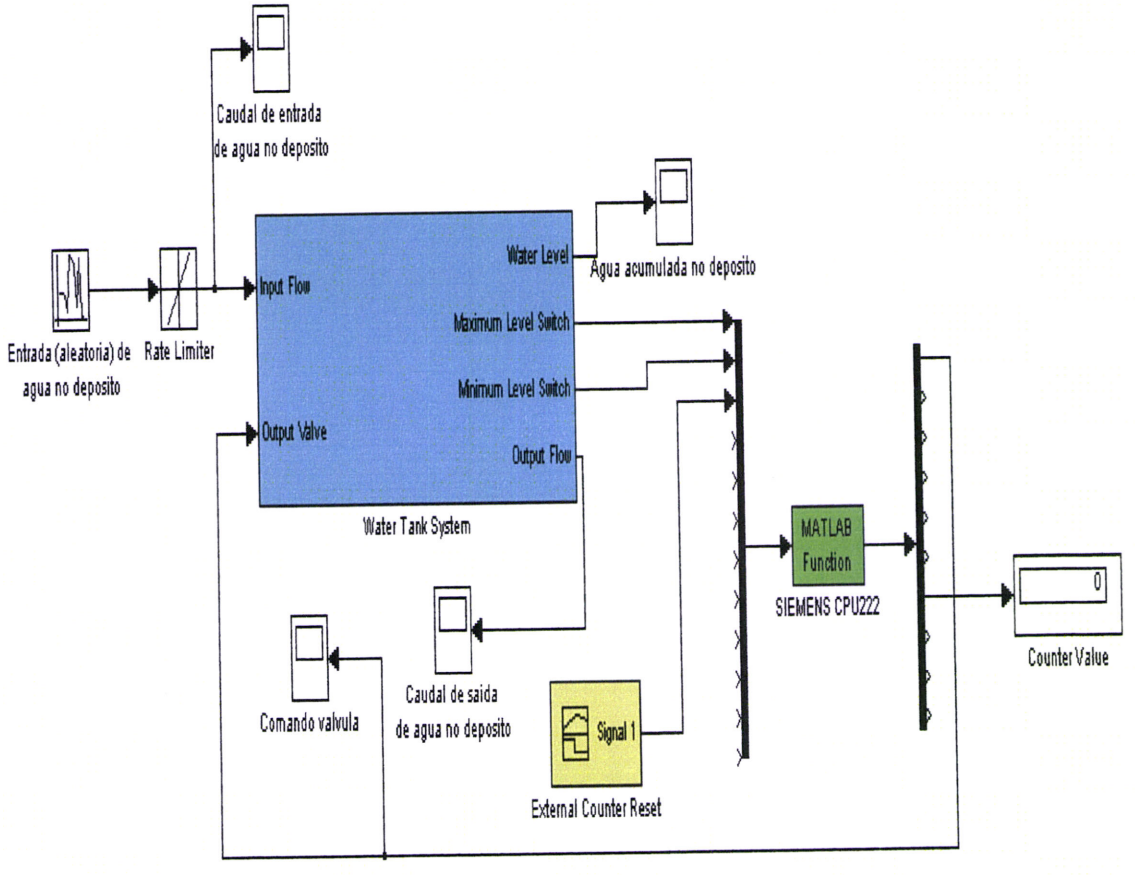


Fig. 20 – Simulação do exemplo do tanque de água

Na fig.21 está representado o interior do bloco azul. O bloco input flow entra o sinal aleatório que corresponde ao caudal que entra para o depósito, em seguida há um somatório entre a entre a entrada de água e a saída dela através de uma válvula que abre e fecha consoante o nível de água no depósito. Depois temos um integrador que vai calcular a diferença de nível de água dentro do depósito. Em seguida encontram se os detectores de nível da água, que vão indicar se devem ou não abrir a válvula consoante o nível que estiver a água. Na segunda e terceira saída do bloco azul é o nível máximo que o depósito pode atingir e nível mínimo que se quer que o depósito atinja, respectivamente. A primeira saída regista graficamente o nível da água e a última saída regista graficamente a saída de água no depósito.

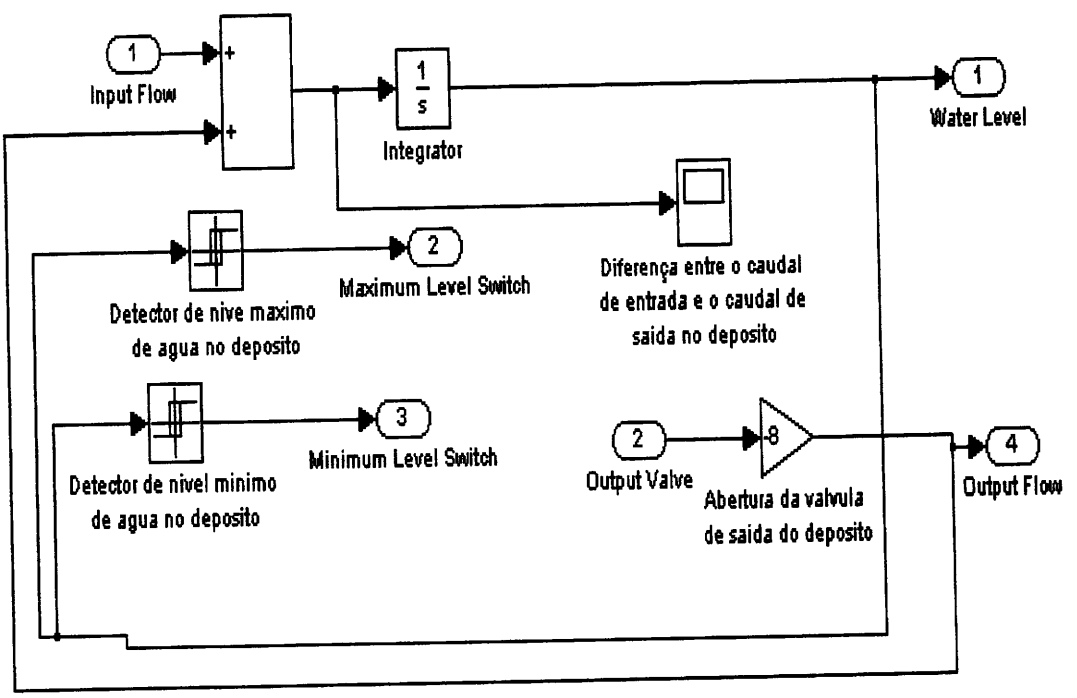


Fig. 21 – Esquema da simulação do controlo do depósito de água.

Como já referido anteriormente em modo genérico, tem-se agora numa forma mais particular a função do nosso autómato. Onde di são entradas digitais e as ai são as entradas analógicas. Esta função é implementada no bloco Matlab Function, desta forma é adicionado no Simulink o programa.

```
function
[saida]=cpu222_3(di1,di2,di3,di4,di5,di6,di7,di8,ai1,ai2,ai3,ai4)

global m0 m0_a m1 m1_a m2 m2_a m3 m3_a m10 m10_a m11 m11_a c1

%inicialização das saidas (pois podem não ser todas utilizadas)
aux=0;
do1=aux;
do2=aux;
do3=aux;
do4=aux;
do5=aux;
do6=aux;
ao1=aux;
ao2=aux;
ao3=aux;
ao4=aux;
%fim de inicialização
```

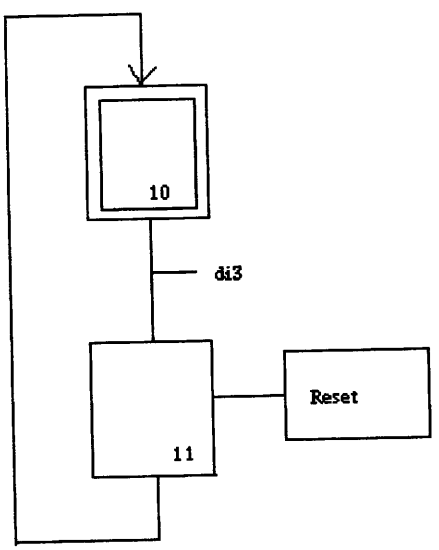


Fig. 22 – Grafcet das etapas do RESET

Para que o processo possa reiniciar, melhor, tornar a contagem a zero (Reset) escreve-se as equações de comando retirados do grafcet anterior, isto é, quando na entrada di3 tiver o sinal 1 bit, acciona o Reset.

```
%Evolução das etapas do grafcet 1
m10=(m11)|(m10&~m11);
m11=(m10&di3)|(m11&~m10);
```

Para contar as vezes que o tanque varia o nível de água, seguesse o seguinte grafcet e dela retira-se as seguintes

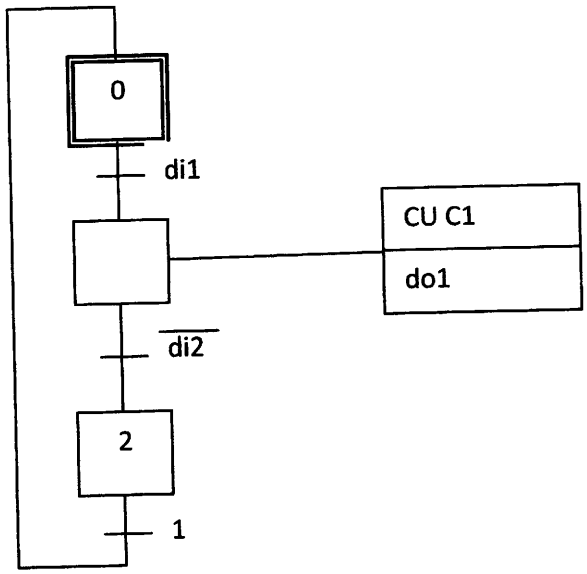


Fig. 23 – Grafcet do contador.

```
%Evolução das etapas do grafcet 2
m0=(m2)|(m0&~m1);
m1=(m0&di1)|(m1&~m2);
m2=(m1&~di2)|(m2&~m0);

%Actualização de contadores
if m1==1 & ~m1_a
    c1=c1+1;
end
%Reset do contador
if m11
    c1=0;
end
```

Para registar a saída do contador tem-se que seleccionar uma saída analógica do autómato. Para que fazer a diferenças de nível de água do depósito, é necessário saber o caudal de saída e de entrada, por isso iguala-se uma saída analógica á primeira memória do contador e á entra do PLC.

```
%Geração das saidas
do1=m1;
ao1=c1;
ao2=ai1;
```

Para que o autómato saiba que etapas irão seguir é preciso indicar e actualiza-las. A memoria m0_a e assim sucessivamente, são as novas etapas que serão as anteriores no próximo ciclo.

```
%Actualizações de estado das várias etapas
m0_a=m0;
m1_a=m1;
m2_a=m2;
m3_a=m3;
```

Para definir as saídas do autómato utiliza-se uma matriz de dimensão 1xm, esta matriz completa a função descrita no inicio deste programa.

```
saida=[do1 do2 do3 do4 do5 do6 ao1 ao2 ao3 ao4];
```

Com isto é terminado o nosso programa.

Á entrada do depósito de água colocou-se um controle de caudal. O resultado está descrito no seguinte gráfico.

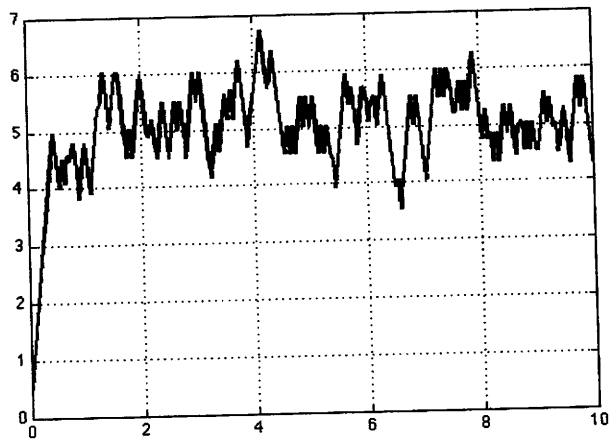


Gráfico 1 – Representação do sinal á entrada do depósito de água em função do tempo.

Neste gráfico pode-se constatar que o caudal é variável. Varia entre 3,5, aproximadamente, e os 7m/s, sendo assim aleatório o seu caudal.

No gráfico seguinte é representado o controlo do nível de água no depósito. Como se pode constatar o nível de água varia os 4 e os 5m de altura, não ultrapassando essa cota.

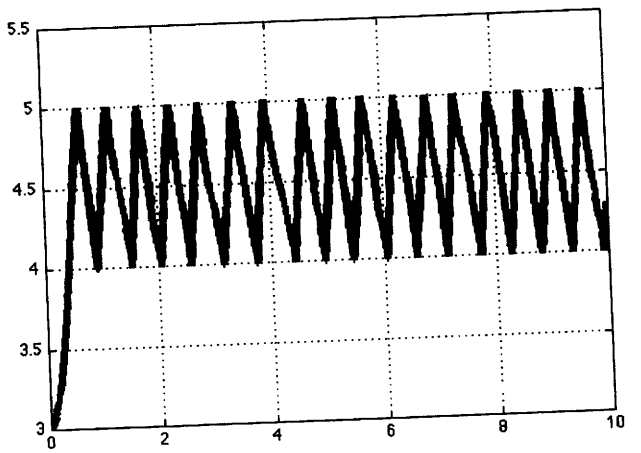


Gráfico 2 – Representação da água acumulada no depósito em função do tempo.

Como isto pode-se comprovar que a simulação no autómato em ambiente Matlab/Simulink é eficaz.

8. Conclusão

Uma nova abordagem para o testar programas controlados por PLC para o ensino de automação foi apresentado. Esta abordagem é baseada no software Matlab / Simulink.

O programa controlado por PLC é traduzido para um bloco de função Matlab, dentro do Matlab / Simulink ambiente, que vão actuar sobre o modelo do processo industrial, desde que a simulação é executada. A tradução desenvolvida traduz automaticamente o programa controlado por PLC, escrito como uma lista de instruções, em linguagem Matlab / Simulink. A tradução produz um ficheiro-m, obtidos através da aplicação de um conjunto de regras que convertem a tradução da lista de instrução PLC em linguagem Matlab. M-Este arquivo é integrado no Matlab / Simulink processo de simulação, em função do bloco denominado "PLC Control Program '.

9. Bibliografia

- ✓ Siemens, Simatic S7-200 Programmable Controller, System Manual, Siemens AG 1998
- ✓ Hut, Brian; Lipsmen, Ronald; Rosemberg, Jonathon – *A guide to Matlab for beginners and experienced users* – Combridge, 1995
- ✓ Herman, Todd; Jones, Allen; MacDonald, Matthew; Rajan, Rakess –*Visual Basic 2008 Recipes* – Apress
- ✓ Mikell P. Groover - *Automation, Production Systems and Computer Integrated Manufacturing* - Prentice-Hall, 1987.
- ✓ Halvorson, Michael – *Microsoft Visual Basic 2008, Step by Step* – Microsoft Press, Washington.
- ✓ Matlab/Simulink, http://www.mathworks.com/.
- ✓ V Pinto, S. Rafael, J: F: Martins; - *PLC controlled industrial processes on-line simulator* - IEEE International Symposium on Industrial Electronics, ISIE 2007, June 2007, Vigo, Spain.
- ✓ Electropedia - electrical and electronic terminology database under the IEC 60050 series (available on-line),http://www.electropedia.org/.
- ✓ Andrew Kusiak - *Computational Intelligence in Design and Manufacturing* - John Wiley & Sons, 2000.
- ✓ S. B. Morriss - *Automated Manufacturing Systems* - McGraw Hill, 1994.
- ✓ A. Rullan, "Programmable logic controllers versus personal computers for process control" - *Computers and Industrial Engineering* 33 - pp. 421-424, 1997.
- ✓ G. L. Kim, P. Paul, Y. Wang - "UPPAAL in a nutshell" - *International Journal on Software Tools for Technology Transfer*, 1, pp. 134-152, 1997.
- ✓ M. Chmiel, E. Hrynkiewicz, M. Muszynski - The way of ladder diagram analysis for small compact programmable controller - Proceedings of the 6th Russian-Korean International Symposium on Science and Technology KORUS-2002, pp. 169-173, 2002.

- ✓ Gi Bum Lee, Han Zandong, Jin S. Lee, "Automatic generation of ladder diagram with control Petri Net", *Journal of Intelligent Manufacturing*, 15, 245±252, 2004.
- ✓ Hyung Seok Kim, Wook Hyun Kwon, Naehyuck Chang; "*A translation method for ladder diagram with application to a manufacturing process*", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 793-798, Detroit, USA, 1999.