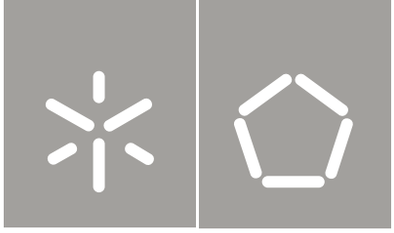
Universidade do Minho
Escola de Engenharia

Carlos David da Silva Barros

Desenvolvimento de Plataformas de Automação Digitais.

Desenvolvimento de Plataformas de Automação Digitais.
Carlos David da Silva Barros
UMinho | 2013

outubro de 2013



Universidade do Minho
Escola de Engenharia

Carlos David da Silva Barros

Desenvolvimento de Plataformas de Automação Digitais.

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao
Grau de Mestre em Engenharia Mecânica

Trabalho efetuado sob a orientação do
Professor Doutor José Mendes Machado

e coorientação da
Professora Doutora Filomena Soares

outubro de 2013

DECLARAÇÃO

Nome: Carlos David da Silva Barros

Correio eletrónico: carlosdavidbarros@gmail.com

Tel./Tlm.: 969431589

Número do Bilhete de Identidade: 13711453

Título da dissertação:

Desenvolvimento de Plataformas de Automação Digitais

Ano de conclusão: 2013

Orientador(es): Professor Doutor José Mendes Machado e Professora Doutora Filomena Soares

Designação do Mestrado: Mestrado Integrado em Engenharia Mecânica

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia

Área de Especialização: Tecnologias de Manufatura

Escola: Escola de Engenharia

Departamento: Departamento de Engenharia Mecânica

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA DISSERTAÇÃO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

Guimarães, ___/___/_____

Assinatura: _____

“O entusiasmo é a maior força da alma. Conserva-o e nunca te faltará poder para conseguires o que desejas.”

Napoleão Bonaparte

RESUMO

A implementação de sistemas automatizados na indústria implica o treino prévio de pessoal especializado na implementação de autómatos programáveis (vulgarmente denominados como PLCs, do inglês *Programmable Logic Controller)*, de sensores e atuadores de vários tipos e funções, desde o simples comando de uma válvula até complexos controladores de processos.

Este trabalho tem como objetivo criar uma ferramenta de simulação, onde os estudantes possam testar a implementação e comportamento de sistemas automatizados reais. Assim esta dissertação apresenta uma plataforma de simulação de sistemas automatizados, réplicas de sistemas reais, para que o comando do sistema possa ser simulado virtualmente, utilizando simulação *Model-In-the-Loop*. A plataforma foi desenvolvida no contexto do ensino de Sistemas a Eventos Discretos a estudantes de Engenharia Mecânica e Engenharia Eletrónica.

A principal vantagem desta plataforma de simulação é o facto da metodologia de desenvolvimento poder ser estendida a outros exemplos práticos ilustrativos, disponibilizando aos estudantes novas estratégias e metodologias de ensino relacionadas com práticas laboratoriais.

No desenvolvimento deste trabalho, procedeu-se à divisão da plataforma de automação em duas partes, parte de comando e parte física. Ambas sincronizadas uma vez que uma não funciona sem a outra. Neste trabalho aborda-se apenas o desenvolvimento da parte de comando sendo a parte de comando abordado noutra trabalho complementar a este.

Dado isto, os passos para o desenvolvimento da parte de comando da plataforma, assim como os formalismos e ferramentas utilizadas estão descritas ao longo desta dissertação.

Palavras Chave:

Ensino de Automação, Simulação de Sistemas Automatizados, Bancadas Didáticas, *Model-In-the-Loop, Software-In-the-Loop, Hardware-In-the-Loop,* SFC, Grafcet, Linguagem *Ladder*.

ABSTRACT

The implementation of automated systems in the industry implies the prior training of specialized personnel in the implementation of PLCs (Programmable Logic Controller), sensors and actuators of various types and functions, from simple command to a valve controlling complex processes.

This work aims to create a simulation tool, where students can test the implementation and performance of automated real. This thesis presents a simulation platform for automated systems, replicas of real systems, so that the control system can be simulated virtually simulation using *Model- In-the - Loop*. The platform was developed in the context of the teaching of Discrete Event Systems to students of Mechanical and Electronics Engineering.

The main advantage of this simulation platform is that the development methodology can be extended to other illustrative examples, providing students with new strategies and teaching methodologies related to laboratory practice.

In developing this work, we proceeded to the division of the automation platform into two parts, the command and the physical. Both synchronized since it will not operate without one another. In this paper only discusses the development of the control command being addressed in another part of this supplementary work.

Given this, the steps for the development of part of the platform control, as well as formalisms and tools used are described throughout this thesis.

Keywords :

Education Automation, Simulation of Automated Systems, Benches Teaching, *Model-In-the-Loop*, *Software-In-the-Loop*, *Hardware-In-the-Loop*, SFC, Grafcet, *Ladder*Language.

AGRADECIMENTOS

Vários foram os fatores intervenientes, ao decorrer deste ano letivo, que contribuíram para a elaboração deste trabalho.

Em primeiro lugar a realização desta Dissertação de Mestrado contou com a ajuda e orientação do Professor Doutor José Machado e Coorientação da Professora Doutora Filomena Soares, os quais com sua disponibilidade, experiência, dedicação, motivação, numerosas sugestões e críticas ajudaram na realização desta, a qual perfaz um ponto fulcral na minha vida académica.

Por outro lado quero agradecer a todos os professores do Departamento de Engenharia Mecânica que ao longo do meu percurso académico me motivaram e incutiram o cada vez maior gosto por esta profissão assim como a vontade de progredir sempre mais.

Quero também agradecer aos meus familiares e amigos o seu apoio, sem o qual não seria possível a realização deste trabalho e deste curso.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FBD – FunctionBlockDiagram

Grafcet – Graphe Fonctionnel de Commande, Etapes Transition

HIL – Hardware-In-the-Loop

HMI – Human/Machine Interface

IEC – International Electronical Commission

IL – Instruction List

LD – Ladder Diagram

MIL – Model-In-the-Loop

PLC – Programmable Logic Controller

RdP – Redes de Petri

SFC – Sequential Function Chart

SIL – Software-In-the-Loop

ST – Structured Text

Capítulo 1
INTRODUÇÃO

1. INTRODUCÃO

1.1. Contexto/Enquadramento

O processo de ensino/aprendizagem tem sofrido alterações [1] desde o Tratado de Bologna: redução das horas de contacto entre o professor e os estudantes e a centralização do aluno no processo de ensino/aprendizagem [2]. Isto implica uma precisa definição dos objetivos e capacidades que o estudante deve adquirir, mas também a reformulação de estratégias e metodologias de ensino. O estudante deve agora ter mais iniciativa e deve ter um papel ativo na sua educação para além do assistir às aulas, escutar o professor e tirar apontamentos.

É sabido que os estudantes retêm melhor o que estão a aprender se o puderem praticar [3]. O conceito "aprender fazendo" é particularmente importante para os estudantes de engenharia, uma vez que terão que se confrontar com situações práticas reais na sua carreira profissional. Portanto, laboratórios e lugares de trabalho para fazer exercícios práticos e pôr em prática a teoria estudada nas aulas são da máxima importância. No entanto, é difícil ter e manter estes lugares de trabalho pois os equipamentos são caros, não há suficiente espaço laboratorial e o pessoal qualificado para ajudar e supervisionar os estudantes não é suficiente [1].

Há também vários laboratórios remotos disponíveis na Internet [4], com aplicações em problemas eletrónicos e controlo de processos, mas ainda existe uma falha no que diz respeito às disciplinas de Sistemas a Eventos Discretos. Por exemplo, não há ainda PLCs (*Programmable Logic Controllers*) disponíveis remotamente para os estudantes testarem os seus programas de comando de problemas reais, desenvolvidos por exemplo em diagramas de *Ladder*.

Para colmatar as debilidades no campo do ensino pode-se ter em conta a simulação em ambientes virtuais e as animações. Desta forma, os estudantes têm a possibilidade de realizar simulações de exercícios práticos quando e onde quiserem, tendo apenas a necessidade de ter um computador pessoal (PC) e o programa correto [1].

A fim de desenvolver um ambiente para o desenvolvimento e teste de problemas de automação é preciso ter em conta que os sistemas automatizados são constituídos por modelos de comando e modelos físicos que trocam mensagens entre si. Por isso é necessário exemplificar o comportamento de comando, o comportamento da parte física e a interação entre os dois modelos, uma vez que estes se são modelizados separadamente e, como referido acima, comunicam entre si.

A interação entre ambos os modelos deve ser modelada como variáveis booleanas e não-booleanas que serão a base para a elaboração dos respetivos modelos.

Formalismos de diagramas de funções sequenciais (SFC) [5] (ou Grafcet), são usados para a especificação da parte de comando dos sistemas. São fáceis de usar, fáceis de aprender pelos estudantes e bem adaptados às especificações de comportamento dos sistemas sequenciais. Depois da formalização da parte de comando do sistema esta especificação pode ser, mais tarde, traduzida em equações algébricas que serão a base para a elaboração do programa em linguagem *Ladder* [6], para ser introduzida no PLC [7].

Para modelar o comportamento da parte física, são utilizados Autómatos Finitos Temporizados. Esta parte é inacessível aos estudantes pois eles só precisam conceber os programas de comando para simular os exercícios. Também, se o modificarem acidentalmente, a simulação funcionará de modo errado e não se comportará como deveria. Este formalismo é altamente adaptável à modelização da parte física pois permite uma abordagem á modelização modelar [8], é um formalismo não determinístico [9] e tem aspetos temporizados formalmente bem definidos e formalizados.

1.2.Objetivos

Com este trabalho pretende desenvolver-se uma ferramenta de simulação onde os estudantes possam testar o comportamento dos sistemas de automação correspondentes a problemas reais, através da conversão das especificações de comportamento do sistema de comando e a visualização das animações do caso em estudo. Tudo isto sem a necessidade das onerosas bancadas físicas que tanta manutenção, espaço e pessoas responsáveis por elas necessitam. Com este trabalho é possível que cada aluno tenha disponível uma bancada virtual e trabalhe nela quando e onde quiser, tudo isto sem a necessidade de esperar pelo horário da aula laboratorial.

1.3.Estrutura da Dissertação

No capítulo 1 é feita a introdução ao trabalho e a apresentação dos objetivos da dissertação.

No capítulo 2 são apresentados vários formalismos de especificação de comando para a modelização do controlador e é efetuada a comparação entre os diversos formalismos analisados

No capítulo 3 apresenta-se o conceito e as técnicas de simulação *Model-In-the-Loop* (MIL), *Software-In-the-Loop* (SIL) e *Hardware-In-the-Loop* (HIL). Também são apresentadas as vantagens e desvantagens de cada uma destas técnicas.

No capítulo 4 aborda-se a metodologia utilizada na realização deste trabalho. Os passos que se tomaram assim como se refere a ferramenta utilizada para a elaboração das plataformas de automação.

No capítulo 5 aborda-se o caso de estudo, descrevendo os passos necessários para se executar a utilização correta na resolução dos problemas. Estão presentes três dos catorze exercícios elaborados. Cada exercício tem o enunciado, o modelo do controlador, os códigos de recetividades e ações utilizados no CX-One, o Grafcet de comando que soluciona o problema, a sua conversão para Ladder assim como a sua implementação no ambiente de simulação.

Por fim, no capítulo 6 são apresentadas as conclusões deste trabalho.

Capítulo 2

FORMALISMOS DE ESPECIFICAÇÃO DO COMANDO

2. Formalismos de Especificação do Comando

Um dos objetivos deste trabalho consiste no desenvolvimento de simulações de sistemas automatizados virtuais, que possam ser utilizados no ensino da automação.

Pretende-se que as simulações virtuais imitem os sistemas automatizados reais, de maneira que a aprendizagem adquirida através da simulação seja a mais parecida possível de um sistema automatizado real. Logo, determinadas metodologias deverão ser idênticas aos sistemas reais.

As simulações são desenvolvidas com o auxílio do programa CX-One, da empresa OMRON, utilizando a linguagem *Ladder*. Este programa permite aos utilizadores criar, configurar e programar dispositivos como por exemplo PLCs (*Programmable Logic Controller*), HMIs *(Interfaces Homem-Máquina*), bem como redes e sistemas de comando de movimento.

2.1. Modelização do Controlador

A parte de comando é realizada por um PLC, este é um controlador lógico programável que se define como um dispositivo eletrónico digital que tem uma memória programável para guardar instruções e levar a cabo funções lógicas de configuração de sequência, sincronização, de contagem e aritméticas, para o controlo de maquinaria e processos. Desta maneira a sua modelização deste autómato consiste na simulação do respetivo PLC (programa e comportamento dinâmico).

Um programa de um PLC pode ser obtido através de um conjunto de expressões booleanas extraídas da interpretação de um formalismo que tenha servido de base à sua especificação. As expressões são avaliadas individualmente em cada "scan", sendo o resultado correspondente armazenado na memória intermédia do PLC. No final de cada execução, a parte de memória correspondente às saídas é copiada [10].

As linguagens de programação utilizadas nos PLCs são normalizados [6], podem ser agrupadas do modo seguinte:

- Linguagens Textuais:
    - Lista de Instruções (Instruction List – IL)
    - Texto Estruturado (Structured Text – ST)

- Linguagens Gráficas:

    - Diagrama de Funções Sequenciais (SFC)

    - Diagramas de Contacto (Ladder Diagram – LD)

    - Diagrama de Blocos de Funções (Function Bloc Diagram – FBD)

Este trabalho surge na sequência de trabalhos anteriores, como por exemplo em [10], daí que algumas escolhas tenham sido efetuadas de acordo com esses trabalhos, de forma a poderem ser comparados resultados obtidos. Assim, o ambiente de simulação escolhido é propriedade do fabricante OMRON [11]. É de salientar que qualquer ambiente de simulação poderia ser utilizado para implementação da metodologia proposta neste capítulo.

A simulação desenvolve-se através do *software* CX-Programmer e CX-Simulator. O programa do controlador (linguagem de programação *Ladder*) é colocado no CX-Programmer, que simula a memória do PLC. Seguidamente, utiliza-se o CX-Simulator para concluir a modelização, simulando o processador do PLC.

Desta forma a linguagem de programação foi pré-definida, pelo uso de *software* CX-One, porém para uma compreensão mais profunda da modelização é necessário selecionar um formalismo para a especificação de comando de sistemas a eventos discretos.

Assim de forma a modelizar o comportamento de sistemas automatizados realizou-se uma análise dos vários formalismos existentes: Autómatos Finitos, Redes de Petri, SFC (IEC 60848), Statecharts e outros.

2.1.1. MÁQUINAS DE ESTADOS FINITOS

As máquinas de estados finitos (ou autómatos finitos) constituem o formalismo de especificação de sistemas a eventos discretos de utilização mais divulgado. A teoria subjacente remonta à década de 50, com as propostas de Moore e de Mealy [12].

Uma máquina de estados finitos é um sistema de transições que possui um número finito de estados e transições etiquetadas entre dois estados a partir da qual é possível descrever a evolução de um sistema ao longo do tempo.

Três modelos podem ser utilizados para a sua representação [13]:

- Teoria dos grafos (recorrendo à teoria de conjuntos);

- Diagramas (modo gráfico);

- Matrizes.

Porém, considerando a legibilidade associada a cada representação, apenas a representação gráfica se considera interessante para a especificação de sistemas.

Dentro das representações gráficas, os diagramas de estados e os fluxogramas são de referência especial. Os primeiros são usados em várias áreas de aplicação enquanto os segundos estão mais dirigidos para a descrição de algoritmos [12].

Os diagramas de estados são constituídos por círculos, que representam os estados e por arcos que representam as transições entre os estados. A figura 1 ilustra um exemplo elementar, sendo S1 o estado inicial.

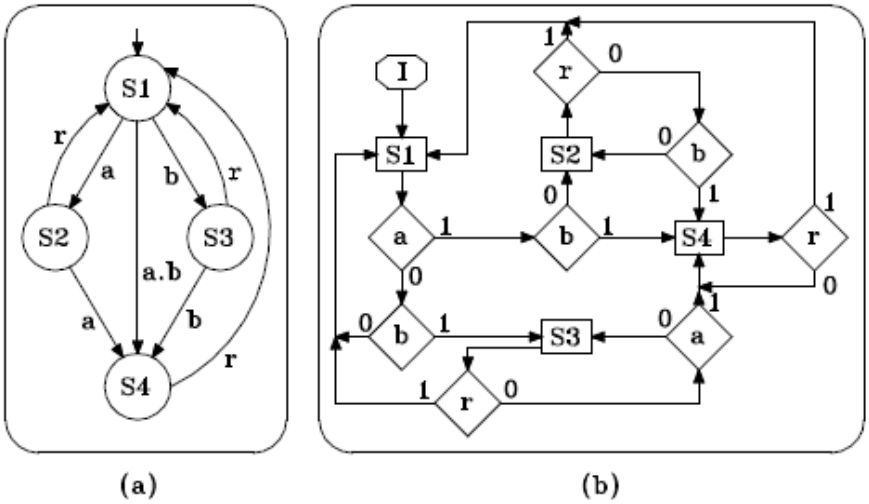


Figura 1 - Máquina de Estados Finitos. a) Diagrama de Estados; b) Fluxograma [14].

A especificação apresentada na figura 1 a), na ocorrência dos eventos a e b, não é executável, pois estão presentes situações de indeterminismo. Uma técnica vulgarmente utilizada para evitar isso, considera a ocorrência mutuamente exclusiva dos eventos especificados ("*a* e não *b*" entre S1 e S2 e " *b* e não *a*" entre S1 e S3).

Os fluxogramas podem considerar-se de compreensão intuitiva. Estes recorrem a representações gráficas em que os estados são representados por retângulos e as transições,

entre estados, são detalhadas em termos das dependências de eventos de entrada, e representadas por losangos, responsáveis pela modelização de testes booleanos.

Enquanto no diagrama da figura 1 a) é possível modelizar situações de indeterminismo, no fluxograma da figura 1 b), garante-se a especificação determinística. A precedência apresentada entre eventos é explicitamente representada, impondo prioridades entre eventos e evitando indeterminismos [12].

De um modo geral, as máquinas de estados finitos apresentam limitações sérias à modelização de sistemas complexos, como por exemplo nas situações de atividades concorrentes e representações hierárquicas tendentes a compactar a representação. Estas dificuldades de modelização podem ser consideradas como uma desvantagem séria das máquinas de elementos finitos e para minimizar estas limitações, várias extensões têm sido propostas. Entre elas, os Statecharts constituem uma resposta adequada a redução destas limitações.

2.1.2. REDES DE PETRI

As Redes de Petri (RdP) foram propostas no ano de 1962 pelo matemático alemão Carl Adam Petri, na sua dissertação de doutoramento.

Estas são um instrumento de modelização e análise de sistemas, mais expressivas que as máquinas de estados finitos. Estas permitem modelizar atividades concorrentes e sincronizações entre processos, exclusão mútua de recursos e memorização. Assim esta ferramenta tem vindo a ser aplicada a várias áreas. Por exemplo, nos sistemas de manufatura, de comando e gestão de recursos e dos sistemas de automação, quando as áreas dos sistemas distribuídos e as questões de comunicação entre os processos são concorrentes [12].

Devido às suas potencialidades de modelização, o desenvolvimento das RdP foi continuado, designadamente em áreas como: sincronização de processos, concorrência, conflitos e partilha de recursos. Até aos dias de hoje têm vindo a ser desenvolvidos trabalhos teóricos e aplicações sobre RdP, tendo este estudo levado, a um desenvolvimento das técnicas de análise das RdP e sua aplicação prática, e ao desenvolvimento de variantes do modelo seminal das RdP tendo em vista aplicações específicas [15]. Dependendo do objetivo em estudo, a sua aplicação poderá ser realizada de diferente modo [14].

Numa primeira abordagem as RdP são consideradas como uma ferramenta auxiliar de análise. Nesta situação, outros formalismos são utilizados para especificar o sistema.

Uma abordagem alternativa à anterior, passa-se por utilizar as RdP para todas as fases do processo de desenvolvimento e tem como objetivo obter uma rede de Petri isenta de erros que será diretamente traduzida para um sistema pronto a operar.

Na primeira abordagem, a ênfase é colocada em transformar uma especificação de sistema na sua representação através da rede de Petri, enquanto na segunda a ênfase é colocado nas técnicas de tradução das RdP que permitirão implementar o sistema a partir da sua representação em RdP.

Uma possível representação das redes de Petri é através de um grafo possuindo dois tipos de nós, designados por lugares (círculos) e transições (barras), como na figura 2. Os lugares são representados por círculos enquanto as transições por barras [15].

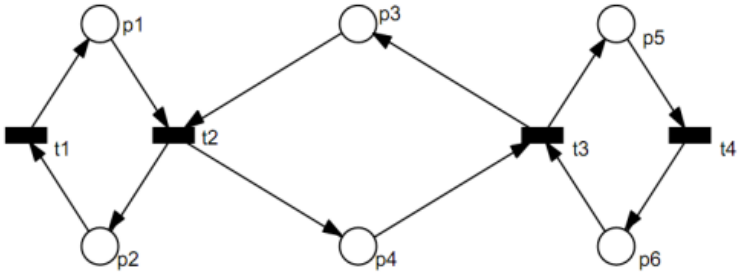


Figura 2 - Exemplo de uma Rede de Petri [14].

Nesta representação é possível associar eventos e condições do sistema que se pretenda modelar às transições e lugares do grafo.

Resumidamente, como ferramentas matemáticas e gráficas, as RdP oferecem um ambiente uniforme para a modelização, análise formal e simulação de sistemas a eventos discretos, permitindo uma visualização simultânea da sua estrutura e comportamento.

2.1.3. GRAFCET (SFC – Sequential Function Chart, IEC 60848)

Baseada nas redes de Petri e desenvolvido em França, na década de 70, fruto dos esforços de académicos e técnicos industriais, que sentiam a necessidade de um formalismo simples, de fácil aplicação e compreensão, o Grafcet tem sido desenvolvido até aos dias de hoje de forma a corresponder aos desafios que são apresentados na modelização da parte de comando de sistemas automatizados, geralmente aplicado a sistemas sequenciais.

Depois de normalizado [5], este formalismo apresenta-se como muito poderoso devido à sua simplicidade de execução e leitura, semântica rigorosa e capacidade de representação elevada. Embora tenha sido preparado visando aplicações eletrotécnicas, o SFC uma vez que descreve funções de comando relativas a determinado sistema independentemente do campo de aplicação, pode também ser aplicado a sistemas não elétricos como sistemas hidráulicos, pneumáticos ou mecânicos.

A linguagem Grafcet permite a fácil comunicação entre profissionais de várias áreas envolvidas no processo de automação.

O diagrama funcional Grafcet possibilita que os comportamentos de um automatismo sejam descritos consoante as informações que este recebe. Este formalismo não pretende minimizar as funções lógicas que representam a dinâmica do sistema. Contrariamente a isso, é na imposição de um funcionamento rigoroso, sem incoerências, bloqueios ou conflitos durante o funcionamento do sistema, que se encontra o seu potencial [5].

As suas principais caraterísticas são [14]:

- Legibilidade e apresentação sintética;
- Facilidade de apresentação;
- Modelização de funções lógicas;
- Modelização da concorrência;
- Oferece uma metodologia de programação estruturada, Top-Down (de forma descendente) que permite o desenvolvimento conceptual do geral para o particular;
- Introduz um conceito de "tarefa" de forma hierarquizada;
- Normalizado.

O funcionamento do automatismo pode ser representado tendo *Etapas,* às quais estão associadas *Ações*; Transições, às quais estão associadas Recetividades; *Ligações Orientadas* que ligam etapas a transições e vice-versa [5].

A dinâmica deste é dada segundo um conjunto de regras que nos permite realizar a evolução seguindo uma lógica ao longo do grafo. Estas regras encontram-se enumeradas e explicadas em [5]. Resumidamente, tem-se as regras da Inicialização, Transposição de uma transição, Evolução das etapas ativas, Evoluções simultâneas e Ativação e Desativação simultânea de uma etapa.

2.1.4. STATECHARTS

Os Statecharts foram desenvolvidos em Israel pelo matemático David Harel e podem ser caracterizados como uma extensão às máquinas de estados finitos, suportando a sua representação hierárquica, utilizando uma notação gráfica muito intuitiva e permitindo modelar evoluções sequenciais e/ou paralelas, incluindo um mecanismo de comunicação global ("broadcast") [12].

Segundo [16] pode então definir-se Statecharts como: "*Statecharts = diagramas de estados + profundidade + ortogonalidade + comunicação global*".

Cada parcela referida sintetiza uma característica fundamental dos Statecharts [12]:

A característica "profundidade" permite implementar uma hierarquia de estados. A cada estado da máquina de estados N, pode estar associada uma máquina de estados N+1 que pode ser ativada ou desativada conforme o estado N esteja marcado ou não respetivamente.

A característica "ortogonalidade", também chamada por decomposição "e", suporta a modelização de atividades paralelas.

Por último, a característica " comunicação global" descreve um mecanismo de comunicação instantânea entre as componentes paralelas do modelo. A comunicação é garantida pela emissão de eventos, associados ao disparo de transições entre estados, que são considerados como eventos de entrada por outras transições.

Na figura 3 apresenta-se um exemplo ilustrando algumas características importantes. O Statechart da figura 3 (a) representa a máquina de estados da figura 3 (b), com ganhos claros, em termos de legibilidade.

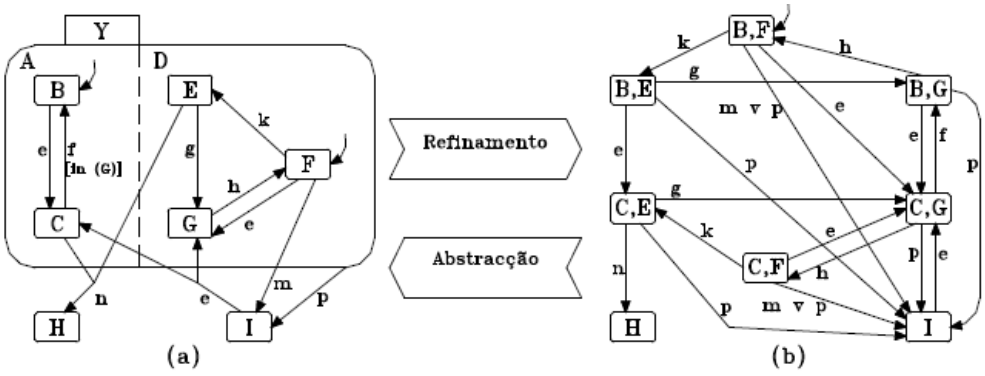


Figura 3 - Exemplo de Statechart (a) e (b) espaço de estados associado [14].

Globalmente, os Statecharts oferecem todas as características necessárias para formalizar uma representação dos sistemas a eventos discretos. São um formalismo de elevada capacidade de modelização que tem como principal característica permitirem uma construção do comando em vários níveis de abstração.

2.2. Vantagens e Desvantagens dos Diferentes Formalismos

Pretende-se que o formalismo utilizado na modelização da parte de comando seja adequado a sistemas sequenciais e que tenha grande capacidade de modelização. Assim, de acordo com a tabela 1, onde estão representadas as vantagens e desvantagens de cada formalismo, foi selecionado o SFC (IEC 60848) [10]. Este embora tenho como ponto fraco a sua dificuldade de aplicação em sistemas de comportamento não sequencial complexos, tem vários aspetos positivos: clareza; notação compacta; fácil aplicação a sistemas paralelos; estrutura gráfica; encontra-se normalizado; é de fácil interpretação e tem uma apresentação sintética.

Tabela 1 - Vantagens e Desvantagens de cada Formalismo [10].

	Vantagens	Desvantagens
Redes de Petri	• Representação Gráfica; • Fácil aprendizagem; • Modelação hierárquica com uma matemática bem definida e fundamento prático; • Abordagem top-down, assim como bottom-up em diferentes níveis de abstração; • Simulável, demonstrável, adequado a projeto.	• Base formal complexa, que não se justifica no estudo de sistemas sequencias de complexidade baixa/média;
Statecharts	• Representação Gráfica; • Hierarquia entre estados; • Ortogonalidade (representação de atividades paralelas); • Interdependência entre estados (mecanismos de comunicação); • Suporte UML.	• Não é a melhor solução para modelação/estudo de sistemas sequencias de complexidade baixa/média;
Autómatos finitos	• Modelo Simples; • Melhores resultados na implementação nas linguagens devido ao grupo restrito de linguagens a que pertence.	• Modelo menos poderoso • Grupo restrito de linguagens • Impossibilidade de modelação de processos paralelos;
SFC (IEC 60848)	• Clareza; • Notação compacta; • Fácil aplicação a sistemas paralelos; • Estrutura Gráfica; • Normalizado; • Interpretação fácil • Apresentação sintética;	• Dificuldade na aplicação em sistemas de comportamento não sequencial complexos;

2.3. Aspetos salientados no Capítulo 2

Estão definidas as regras sistemáticas para a modelização de controladores de qualquer sistema automatizado. Depois de modelizado, um sistema pode ser convertido para linguagem *Ladder*, através da modelização algébrica do Grafcet, obtendo-se um conjunto de equações que podem ser implementadas seguindo regras pré-definidas.

A modelização da parte de comando é feita em função da sequência de acontecimentos que o sistema terá que cumprir (problema), como se de um PLC (ou outro controlador) se tratasse, bastando para isso transferir para o respetivo PLC o programa referente ao problema (em linguagem *Ladder*).

Capítulo 3
SIMULAÇÃO

3. Simulação

Segundo Shanon em [17]: "*Simulação é o processo de elaborar um modelo de um sistema real e conduzir experiências com esse modelo tendo como propósito a compreensão do comportamento do sistema ou a avaliação de diversas estratégias (dentro dos limites impostos por um critério ou conjunto de critérios) para a operação do sistema*". Ou seja a simulação é uma técnica de análise que representa um sistema recorrendo a um modelo computacional para compreender o seu comportamento e perceber se corresponde aos requisitos da especificação de comando [18].

Esta ferramenta consiste na modelização de um sistema real, simulando as condições reais de operação. Para realizar uma simulação é necessário, construir teorias e hipóteses considerando observações efetuadas ao sistema, usar modelos para a descrição/especificação do comportamento do sistema. Estas suposições, que normalmente tomam a forma de relações lógicas ou matemáticas, constituem o modelo [10]. Traduzindo-se desta forma na importação da realidade para um ambiente controlado onde se pode estudar, como referido anteriormente, o comportamento do mesmo, sob diversas condições, sem riscos físicos e/ou grandes custos envolvidos. Com base na criação nesses modelos, são realizadas observações e inferências nas características de operação do sistema real representado [19].

As primeiras linguagens específicas para a simulação surgiram na década de 60. Facilitando ao utilizador a transformação do modelo formal do sistema num programa computacional e disponibilizando funções destinadas a amostragens, análises estatísticas e comando do avanço do tempo na simulação. Esta técnica é por isso uma das mais poderosas ferramentas de análise disponível para o projeto e a operação de processos ou sistemas. Esta poder ser útil desde a fase de análise do problema e definição de requisitos, até às fases de projeto, justificação e implementação e operação. Podendo portanto ser utilizada por todo o ciclo de vida de um sistema automatizado [20].

Nos últimos anos assistimos ao aumento da complexidade dos sistemas automatizados, devido ao cada vez maior uso destes na indústria. Por isso, a construção deste tipo de sistemas torna-se numa tarefa cada vez mais árdua, requerendo o esforço de vários engenheiros [21]. Tornou-se também necessário que as linguagens de simulação para além de nos conduzir a resultados confiáveis, mostrassem que os seus benefícios eram reais. Surgiram então as animações, *software* acoplado aos simuladores, capazes de representar o funcionamento dos sistemas

graficamente. Assim a demonstração de resultados ao utilizador tornou-se muito mais fácil. Este tipo de *software* acabou por proporcionar várias vantagens, entre elas a formação de pessoal [20].

Por razões económicas, os autómatos devem funcionar como planeado assim que forem postos à prova. Por isso devem ser efetuadas tarefas que assegurem que o trabalho está sem atraso e encontra-se bem elaborado.

O métodos mais convencional assegurar o antes referido seria elaborar o teste manual dos sistemas automatizados, porém devido ao aumento da complexidade destes sistemas, o método manual atingiu o seu limite e não consegue cobrir todas as possíveis falhas do sistema [21].

O novo método é contruir e simular modelos computorizados da parte física do sistema automatizado antes de o pôr em serviço. Erros de projeto (componentes inadequados), falhas nos componentes (colisão de cilindros, ventosas de aspiração inadequadas), falhas do programa de comando ou o acontecimento de estados críticos do sistema físico podem ser identificados e resolvidos/tratados antes que o sistema seja instalado [21] e até durante a sua fase de projeto [22]. A utilização da simulação permite assim testar um sistema de uma maneira mais rápida, a um custo mais reduzido do que utilizando testes práticos [23]. A segurança no desenvolvimento de protótipos é também aumentada uma vez que quando o sistema é testado a probabilidade da existência de erros, que não tenham sido detetados na simulação, é mais reduzido do que os erros inerentes a um sistema físico que tenha sido montado sem antes prever o seu comportamento.

- O Papel dos Computadores na Simulação

A utilização da simulação no ambiente empresarial tem-se tornado viável com a evolução da informática. Com o poder computacional dos computadores pessoais a simulação de problemas mais complexos tornou-se computacionalmente possível. Esse facto impulsionou o avanço das aplicações de simulação que hoje apresentam soluções dos mais variados tipos.

O s*oftware* com animação gráfica possibilita que a simulação se associe a ferramentas visuais para tomada de decisão. As interfaces mais amigáveis e as linguagens de programação menos

complicadas também têm contribuído para a popularização da simulação como uma técnica de apoio à decisão.

Assim, com a consolidação das plataformas gráficas, a simulação tem vindo a tornar-se numa ferramenta imprescindível em áreas tais como a investigação, o desenvolvimento de novos produtos entre outros [20].

- Conceitos associados à Simulação

Para se entender melhor o que é a simulação, precisa-se conhecer também as definições de sistemas e modelos. Um sistema é um conjunto de elementos distintos, que exercem entre si uma interação ou interdependência [20]. Por natureza, os sistemas são limitados, ou seja, deve-se definir limites ou fronteiras. Portanto, pode-se definir sistemas dentro de outros sistemas, e assim por diante. Um modelo, segundo Hillier [24], é uma representação de um sistema real, na qual somente os aspetos relevantes para uma determinada análise deste sistema são considerados.

Existe, para além da definição proposta no início deste capítulo, um grande número de definições para a simulação como se pode verificar em vários livros clássicos sobre o assunto tais como os proporcionados nas referências [24] a [30].

Resumidamente a simulação é uma técnica de análise que consiste em representar um sistema recorrendo a um computador para entender o seu funcionamento e perceber se corresponde aos requisitos para que foi projetado, envolvendo assim a modelização de um sistema real baseado em condições reais de operação [10]. O objetivo desta ferramenta passa sempre pelo mesmo: poupar dinheiro e tempo na fase de desenvolvimento do produto, evitar danos materiais e aos trabalhadores, assegurando que o sistema se comportará como planeado antes que seja contruído e posto à prova [21].

Neste trabalho são abordados sistemas automatizados e a problemática da sua simulação.

Um sistema automatizado é, como vimos anteriormente, sempre composto por duas partes, figura 4, a parte física (parte controlada ou processo) e a parte de comando (que controla a parte física), conectadas para que as instruções e informações sejam enviadas do comando para a parte física e vice-versa.

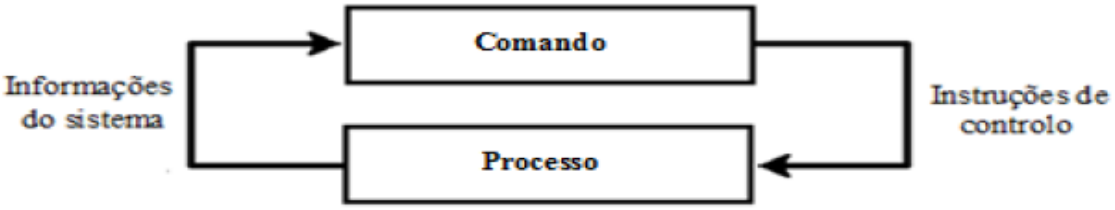


Figura 4 - Sistema Automatizado e suas partes constituintes (parte de comando e parte física)

A simulação pode ser então efetuada de três formas: *Model-in-the-loop* (MIL), *Software-in-the-loop* (SIL) e *Hardware-in-the-loop* (HIL) como representado na figura 5.

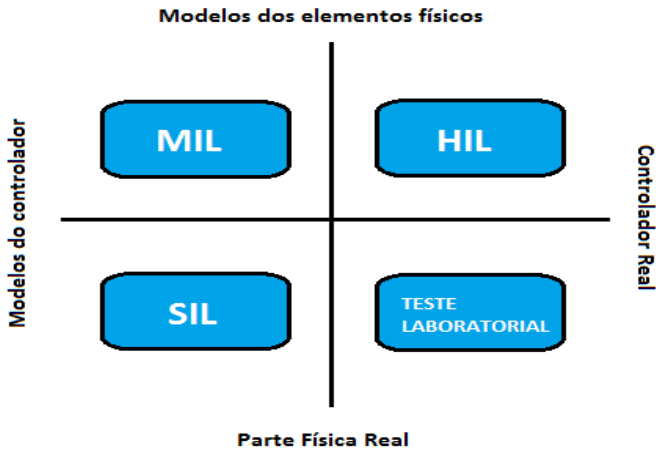


Figura 5 - Diferentes Técnicas de Simulação MIL, SIL e HIL.

3.1. Simulação *Model-in-the-loop*

A simulação *Model-in-the-loop* (MIL) é a simulação onde existe interação entre o controlador modelizado e parte física modelizada sem componentes reais à mistura, como podemos ver na figura 5. Normalmente o modelo do controlador e a parte física estão ligados num circuito fechado [21].

Ambas as partes do sistema automatizado (controlador e parte física) podem ser avaliados e desenvolvidos através do uso da simulação MIL e obter resultados da simulação que ajudam a compreender o comportamento do sistema no seu funcionamento real [31].

Esta simulação oferece um risco reduzido na simulação de diferentes técnicas e metodologias de comando, e como foi referido acima, sem a necessidade de componentes reais. Tornando

assim o investimento monetário muito reduzido e evitando acidentes materiais ou humanos procedentes de erros no projeto inicial dos controladores, permitindo identificar, corrigir e eliminar esses erros.

Esta ferramenta versátil permite a rápida e simples validação de diversos algoritmos, e permite a comparação detalhada entre diferentes estratégias para resolver uma certa tarefa. Desta forma a simulação MIL é amplamente utilizada na fase inicial do desenvolvimento de projetos de controladores, utilizados para o comando de sistemas reais. Ao longo da simulação, o modelo do controlador comunica com o modelo da parte física, enviando sinais com ordens para os modelos dos atuadores, recebendo sinais com informação proveniente da parte física modelada (sinais de sensores, por exemplo) porém o facto de esta técnica ser totalmente virtual traz várias desvantagens como a sincronização dos modelos e trocas de mensagens entre eles. Os resultados são especialmente significativos pois o ambiente de simulação permite testar cada estratégia sob as mesmas condições.

3.2. Simulação *Software-in-the-loop*

A simulação *Software-in-the-loop* (SIL) é a técnica de simulação onde existe interação entre o controlador modelizado e parte física real [10], figura 5, ou modelizada com componentes reais adicionados à simulação (sensores, cilindros pneumáticos) enquanto certos componentes são emulados em computador. Isto torna óbvio que a simulação SIL pode ser facilmente obtida a partir da simulação MIL. A SIL combina a flexibilidade e o baixo custo da simulação, com a fidelidade de um emulador de *hardware*. Pode ser facilmente utilizada na conceção do projeto e nas fases de teste dos programas, sendo por isso capaz de oferecer uma redução significativa dos custos, maximizando a reutilização de código e minimizando o esforço do seu desenvolvimento [32].

Normalmente o modelo do controlador e a parte física estão ligados num circuito fechado [21].

Tal como na MIL esta ferramenta versátil permite a rápida e simples validação de diversos algoritmos, e permite a comparação detalhada entre diferentes estratégias para resolver uma certa tarefa. Os resultados são especialmente significativos pois o ambiente de simulação permite testar cada estratégia sob as mesmas condições [33].

Numa configuração SIL toda a bancada de simulação pode encontrar-se num ambiente controlado, localizado num laboratório. Vários aparelhos, localizados no próprio laboratório ou fora deste, podem ser ligados simultaneamente ao computador onde o simulador está a ser executado através de uma rede *wireless* ou de outro tipo semelhante. Isto permite que sejam possíveis realizar um variadíssimo leque de testes que não seriam possíveis com as formas mais tradicionais de simulação, tais como testes de campo em tempo real ou testes em sistemas de maior complexidade.

Tal como na simulação MIL, uma vez que é possível elaborar esta simulação sem o uso do controlador real, é possível poupar investimentos e evitar acidentes perigosos resultantes de erros no projeto inicial dos controladores, permitindo assim a identificação e eliminação desses erros. O uso da simulação SIL, para as diferentes regiões de operação, incluindo modelos de erro, permite a seleção das estratégias adequadas de comando na plataforma real, pois facilita a repetição de testes de desempenho [10].

3.3. Simulação *Hardware-in-the-loop*.

A simulação Hardware-in-the-loop (HIL) refere-se a uma técnica de simulação que mistura tanto elementos virtuais como reais [34] onde o *software* de comando corre no controlador real e é criado o modelo virtual da parte operativa que interage com o controlador real [10], [21], figura 5. A simulação monitoriza os sinais de saída do sistema em teste (ordens para os atuadores e informação operacional para os *displays*) e injeta sinteticamente sinais de entrada gerados (sinais provenientes dos sensores e comandos provenientes dos atuadores) nos *timings* apropriados. Assim sendo, as saídas do sistema servem como entradas para a simulação, e esta gera saídas que irão ser as entradas do sistema [35].

Como referido anteriormente na parte operativa podem existir também componentes reais como sensores enquanto outros componentes podem ser emulados no computador, correndo como modelos da parte física. Torna-se então óbvio que a simulação HIL pode ser facilmente obtida a partir da simulação SIL [21], [36].

Uma vez que é utilizado um controlador real na simulação e podem ser incluídos também componentes físicos reais, isto permite aproximar a experiência a um cenário mais real. Desta forma a simulação HIL permite a simulação de vários cenários que podem ser muito caros ou difíceis de traduzir num protótipo [23].

Esta característica torna a simulação HIL numa ferramenta bastante útil para a avaliação e desenvolvimento de controladores, oferecendo um risco nulo na experimentação de diferentes técnicas e metodologias de comando. Desta forma, tal como as simulações anteriores, é possível poupar investimentos e evitar consequências perigosas de erros no projeto inicial dos controladores, permitindo assim a identificação e eliminação desses erros [10].

O método de simulação HIL é mais dispendioso que os anteriormente descritos, pois é necessária a utilização de controladores físicos.

3.4. Vantagens e Desvantagens do MIL, SIL e HIL

As simulações *Model-in-the-loop*, *Software-in-the-loop* e *Hardware-in-the-loop* são abordagens importantes que permitem criar analogias do comportamento de sistemas automatizados. Todas estas abordagens permitem testar funções dos comportamentos dos sistemas simulados sob determinadas condições.

Como vimos anteriormente, estes tipos de simulação têm a vantagem de serem seguros, evitarem acidentes materiais ou pessoais perigosos resultantes de erros no projeto inicial dos controladores, permitindo assim a identificação e eliminação desses erros [10]. Além disso o tempo de análise de validação e identificação de erros durante o desenvolvimento de novos produtos é bastante reduzido que com a elaboração de testes manuais [21], [22].

Comparando os métodos, todos eles têm vantagens e desvantagens. A simulação MIL tem a vantagem de ser executada na íntegra através de *software* e modelização dos modelos físicos, o que torna o seu custo substancialmente reduzido. No entanto, um obstáculo a esta interface é o dimensionamento do tempo das trocas de mensagens de saída e entrada dos componentes que podem ser executados em tempo real e a sincronização dos modelos [10], [31].

Na simulação SIL tal como na simulação MIL, o controlador é emulado em computador o que reduz o custo, porém como podem haver componentes reais na parte de comando, esta nunca terá o custo tão reduzido como na simulação MIL. Comparativamente com outras abordagens, a simulação SIL é uma ferramenta que permite realizar facilmente testes de simulação em ambientes controlados com grande flexibilidade, repetibilidade, a grandes velocidades de processamento (permitidas pela evolução dos processadores contidos nos computadores), sendo assim possível obter resultados bastante fiáveis num intervalo de tempo reduzido.

No entanto, possui também algumas desvantagens. Muitos simuladores, principalmente os mais económicos, não possuem a capacidade de simular com rigor o tempo real. Isto pode ser um problema para simulações em que o tempo seja importante, podendo levar a uma discrepância nos resultados obtidos na simulação e o que acontece na realidade. Outro fator importante é o facto de que com o aumento da complexidade dos sistemas e/ou dos testes realizados sobre estes, são necessários computadores e *software* mais potentes. Estes normalmente têm custos bastante elevados, sendo por isso inacessíveis a empresas que não tenham capacidade de realizar este tipo de investimento.

A simulação HIL tem como vantagem que apenas a parte física é simulada e a parte de comando é, como vimos anteriormente, executada no controlador real, assim como pode haver a integração de componentes reais na simulação da parte física [21].Comparada com a abordagem SIL, a abordagem HIL possui algumas vantagens devido ao fato da simulação ser realizada com um programa real a correr no controlador real. Isto pode ser importante, especialmente, quando são consideradas diferentes escalas de tempo de um ponto de vista real, sobretudo no que diz respeito da evolução do controlador real e a sua escala de tempo, correndo nas condições de trabalho reais. Desta forma o erro referente à simulação é reduzido e a simulação torna-se mais fidedigna [10]. Por outro lado, a necessidade de um controlador real causa um aumento do custo da simulação e a interação entre o controlador real e o modelo da parte física, que ocorre no simulador do computador, é ainda complexa e difícil de ser executada, principalmente quando a simulação e o comportamento em tempo real são críticos para a garantia dos resultados obtidos para o sistema testado [37].

3.5. Aspetos Salientados no Capítulo 3

Para este trabalho foram elaborados várias plataformas de simulação de vários sistemas automatizados para que os alunos pudessem ter na ferramenta desenvolvida uma ferramenta de ensino de sistemas a eventos discretos. Nesta poderão ser feitos vários comandos de maneira que através da técnica da análise os distintos comportamentos do sistema possam ser observados.

Desta forma, como referido anteriormente, a definição presente em [17]: "*Simulação é o processo de elaborar um modelo de um sistema real e conduzir experiências com esse modelo tendo como propósito a compreensão do comportamento do sistema ou a avaliação de*

diversas estratégias (dentro dos limites impostos por um critério ou conjunto de critérios) para a operação do sistema", mostra claramente como a simulação é abordada neste trabalho.

Existem três diferentes técnicas de simulações: *Model-in-the-loop*(MIL), *Software-in-the-loop* (SIL) e *Hardware-in-the-loop* (HIL). Cada uma é mais adequada que outra a certas situações, e cada uma tem a sua situação própria em que tem que ser aplicada.

O custo referente ao desenvolvimento da simulação de um determinado sistema é substancialmente inferior ao desenvolvimento de um protótipo do mesmo, ou seja constituído por elementos reais. Porém a simulação não pode substituir na íntegra as tarefas de um sistema real, pois na simulação trata-se com a análise modelos da realidade, logo não se realizam tarefas que o mesmo executa. Contudo a simulação pode ajudar na melhoria do *software* de comando do sistema durante o desenvolvimento do projeto e evita erros de conceção, tornando os controladores e sistemas automatizados industriais mais confiáveis.

Capítulo 4
METODOLOGIA UTILIZADA

4. Metodologia utilizada

A bancada de simulação, está dividida em duas partes, uma parte de comando e uma parte física. Ambas as partes, quando em separado não são significativas. Isto pois uma é complementar à outra e uma não funciona sem a outra. Quando juntas possibilitam o funcionamento dos sistemas automatizados.

A parte de comando é, como diz o seu nome, o comando do sistema, o que dita como o sistema tem que funcionar, o que fazer e em que momento, este atribui ordens ao sistema e recebe deste informação para saber em que estado este se encontra.

A parte física representa os componentes presentes no sistema automatizado (cilindros, motores, ventosas, sensores, entre outros). Neste trabalho, optou-se pela utilização da técnica de simulação MIL por esta permitir a utilização de ambas as partes da plataforma de simulação virtuais intercomunicantes, possibilitar a compreensão do comportamento do sistema no seu funcionamento real e permitir comparar diferentes estratégias para resolver uma tarefa. Dado isto a parte física é virtual, ou seja os componentes do sistema estão modelizados virtualmente de maneira a que se comportem como sistemas reais. Para cada componente do sistema existe um modelo virtual do respetivo componente. Esta parte do trabalho foi desenvolvida no âmbito de outro trabalho complementar a este [38].

Este tipo de técnica permite-nos então realizar uma ferramenta pedagógica onde o utilizador possa testar comportamentos de sistema automatizados reais, sem a necessidade de componentes reais – o que possibilita a elaboração de plataformas mais baratas e mais seguras – e permite que as simulações imitem os sistemas reais, de forma que a aprendizagem adquirida seja a mais parecida possível com a aprendizagem adquirida através de um sistema real.

No desenvolvimento das plataformas de simulação dos sistemas automatizados foi então elaborada para cada plataforma proposta, uma parte comando e uma parte física virtual com os modelos de cada componente da plataforma proposta. Os modelos depois de elaborados foram transcritos para o programa CX-One, onde se pode programar o seu comportamento através de linguagem de programação *Ladder* (utilizando o subprograma CX-Programmer), e fazer o ecrã virtual de maneira a que o funcionamento do sistema seja observável (utilizando o subprograma CX-Designer).

Para evitar a dessincronização de ambos os modelos do sistema automatizado optou-se por colocar ambos os modelos no mesmo ficheiro de programação. Esta estratégia permite que a cada "scan" da parte de comando seja feito um "scan" da parte física, evitando assim que a troca de mensagens entre os modelos seja demasiado lenta ou demasiado rápida.

Este capítulo refere-se à elaboração da parte de comando da bancada de simulação. Assim para esta tarefa é necessário realizar diversas sub-tarefas como o estudo das características do sistema, qual o seu funcionamento ideal, a construção de um Grafcet, a utilização correta do ambiente de estudo, a passagem de SFC para linguagem de programação *Ladder* e sua passagem para o ambiente de trabalho.

Como referido anteriormente, um dos objetivos deste trabalho é a elaboração de uma metodologia padrão para a construção da parte de comando, que possa ser seguida por qualquer pessoa que queira construir uma bancada de simulação, a partir do enunciado de um exercício.

Seguindo a metodologia, pré-definida proposta no sub-capítulo 4.2 é possível reduzir o tempo despendido na formulação e resolução do problema, minimizando a ocorrência de erros na construção do programa de comando da bancada de simulação.

4.1. Ambiente de Trabalho

Pelo facto de as linguagens de programação utilizadas neste trabalho estarem normalizadas, poderia ter sido utilizada qualquer ferramenta de trabalho. Porém devido à existência de um acordo de colaboração entre a empresa OMRON e a Universidade do Minho, vigente já para diversos projetos anteriores, os PLCs disponíveis são da empresa OMRON pelo que se utilizou neste trabalho o programa CX-One da empresa OMRON.

Dado que este programa utiliza como linguagem de programação a linguagem *Ladder*, esta foi a abordada na metodologia de trabalho.

Para a elaboração das partes de comando de cada exercício foi utilizado o formalismo de comando SFC [5].

4.1.1. CX-One

Como referido anteriormente, para realizar as simulações utiliza-se o CX-One (CX-Programmer, CX-Simulator e o CX-Designer) como ambiente de trabalho.

Estas simulações são elaboradas através da modelização da parte física dos sistemas automatizados, ou seja, através de uma base de dados, onde estão modelados alguns dos elementos pertencentes a sistemas automatizados:

- Pneumáticos
- Hidráulicos
- Eletropneumáticos
- Eletro-hidráulicos
- Elétricos

O desenvolvimento das simulações consiste na junção das modelações dos vários componentes dos sistemas em causa, sejam eles qualquer um dos acima descritos.

A modelização dos sistemas é convertida em linguagem *Ladder* e em seguida, o programa desenvolvido em linguagem *Ladder* é colocado no CX-One (CX-Programmer).

O CX-One permite aos utilizadores criar, configurar e programar um conjunto de dispositivos, como, por exemplo, PLCs, HMIs (*Human/Machine Interface*), bem como redes e sistemas de comando de movimento utilizando apenas um pacote de *software* com um número de licença e instalação. Deste modo, a complexidade da configuração é significativamente reduzida permitindo a programação ou configuração de sistemas de automatização com uma formação mínima [10], [39].

Nos Anexos A e B encontra-se descrito detalhadamente como se utiliza cada um dos sub-programas do CX-One (CX-Programmer e CX-Designer) na elaboração das simulações.

4.1.1.1. CX-Programmer

O CX-Programmer oferece uma plataforma de *software* para todos os tipos de controladores PLC da OMRON – desde os micro PLC até aos sistemas de processamento Duplex. Isto permite uma conversão e reutilização fáceis do código PLC entre diferentes tipos de PLC e a reutilização completa de programas de comando criados por gerações mais antigas de *software* de programação PLC.

Na figura 6 é apresentada a janela principal do CX-Programmer, mostrando as suas principais características.

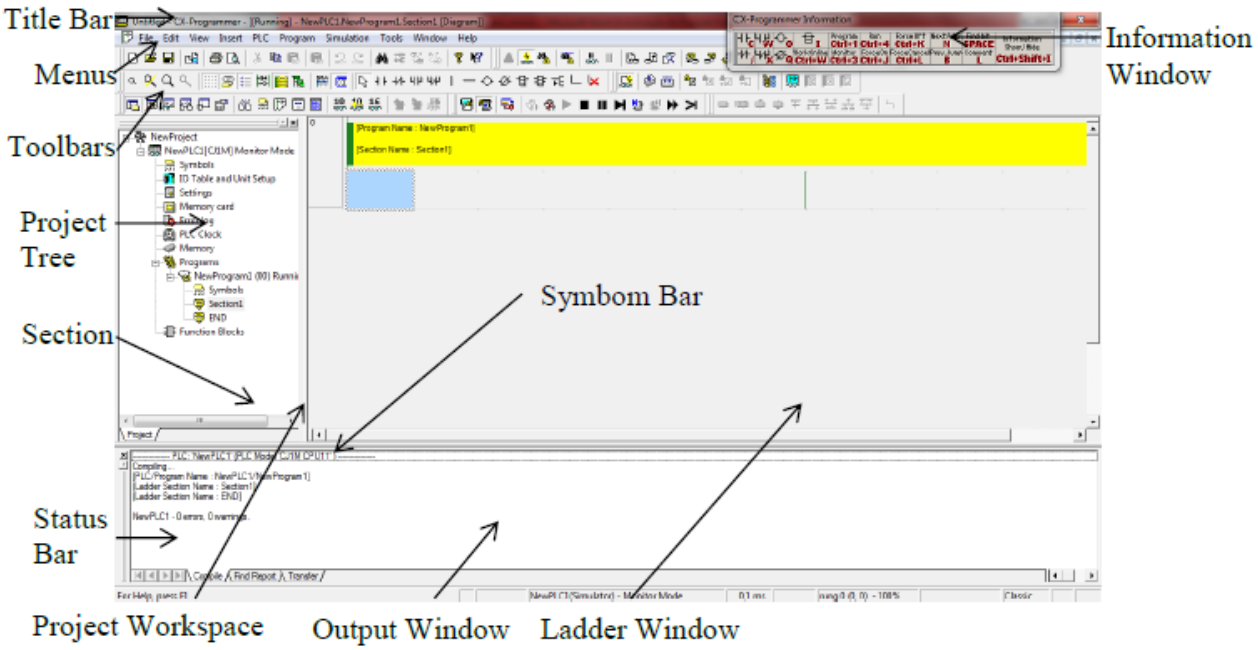


Figura 6 - Janela principal do CX-Programmer [10]

O CX-Programmer está totalmente integrado no pacote de *software* CX-One distribuído pela OMRON, apoiando-se na programação de blocos de funções *standard* programadas em texto estruturado ou numa linguagem em *Ladder* convencional [40].

Os blocos de funções utilizam uma linguagem de programação semelhante ao *Basic*, facilitando tarefas de processamento numérico ou comparações lógicas complexas, necessitando apenas de algumas linhas de código, que podem ser feitas em poucos minutos [10].

O CX-Programmer torna o desenvolvimento de programas numa configuração de arrastar e largar "Drag & Drop" e apresenta as seguintes características:

- Integrado em CX-One, pacote de *software* universal da Omron;
- Ligação automática através de USB ou ligações série;
- Segurança avançada: proteger o conhecimento profundo de propriedade;
- Ecrãs de configuração acessíveis para todas as unidades PLC;
- Ferramentas de simulação PLC incluídas: teste antes de efetuar a transferência.

4.1.1.2. CX-Simulator

Utilizando o CX-Simulator é possível obter um ambiente de execução do programa e análise de defeitos equivalente ao ambiente do sistema PLC real através da simulação do funcionamento de um PLC da série CS/CJ com um PLC virtual no computador.

Este permite a análise de defeito e execução do programa num único PLC antes da montagem do sistema real e reduz o tempo inicial total necessário para o arranque e desenvolvimento da máquina/equipamento [41].

Este programa apresenta as seguintes características:

- Integrado em CX-One, pacote de *software* universal da Omron;
- É possível utilizar todas as funções de depuração disponíveis no CX-Programmer;
- É possível verificar o tempo de ciclo sem o sistema PLC real;
- Executar operações de depuração eficazes que não possam ser executadas no PLC real, como, por exemplo, executar passos e ciclos individuais e inserir interrupções;
- É possível utilizar vários métodos que permitem criar e reproduzir entradas externas virtuais.

4.1.1.3. CX-Designer

CX-Designer trata-se do *software* HMI (interface homem-máquina) utilizado na série NS, podendo também verificar a operação dos dados de ecrã criados no computador.

"*O CX-Designer permite a criação, simulação e lançamento de projetos de ecrã. Os utilizadores podem desenvolver ecrãs mais eficientemente com este software de suporte fácil de utilizar. Este software possui cerca de 1500 objetos funcionais padrão com os gráficos e*

funções avançadas associados, por isso, mesmo utilizadores inexperientes podem criar facilmente ecrãs, apenas ao dispor objetos funcionais num ecrã" [10].

Na figura 7 é apresentada a janela principal do CX-Designer, mostrando as principais características.

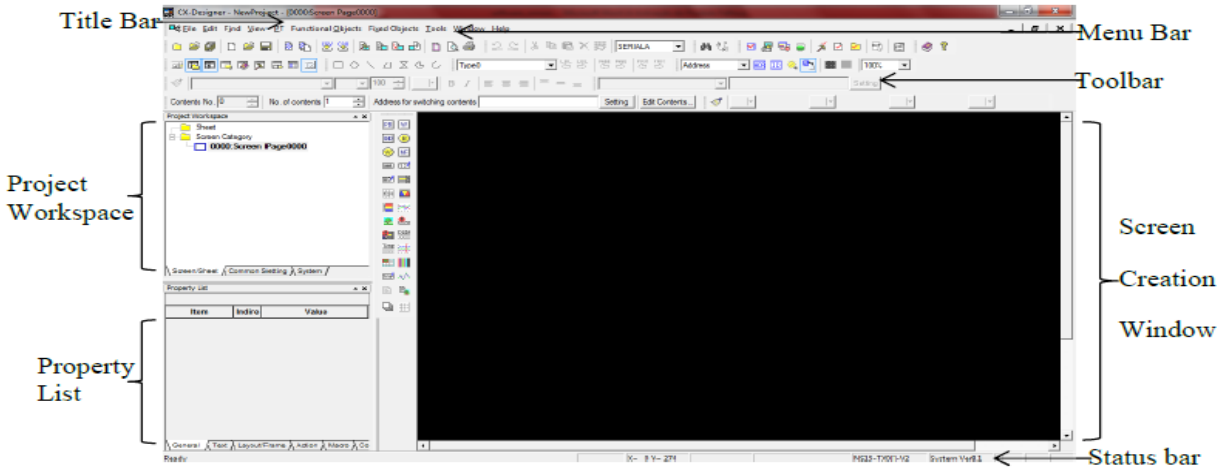


Figura 7 - Janela principal do CX-Designer [10]

Basicamente, este sistema substitui as consolas (por exemplo botoneiras e visores) utilizadas nos sistemas automatizados assim, o estado em que o sistema se encontra durante a simulação deste poder ser analisado mais facilmente [10], [42].

4.1.1.4. CX-Supervisor

O CX-Supervisor destina-se à conceção e à operação da visualização de controladores programáveis e do comando de máquinas. Para além de ser fácil de utilizar em pequenas tarefas de supervisão e comando, ele também oferece uma vasta capacidade para a conceção das aplicações mais sofisticadas.

O CX-Supervisor melhora funções potentes para uma ampla gama de requisitos HMI baseados no Controlador programável. Podem ser criadas aplicações simples com a ajuda de um número de funções e bibliotecas pré-definidas e mesmo as aplicações complexas podem ser geradas com uma linguagem de programação potente. O CX-Supervisor tem uma utilização extremamente simples e intuitiva [43].

4.2. Elaboração do Programa de Comando

Neste subcapítulo descreve-se como elaborar a parte de comando das plataformas de simulação desenvolvidas nesta dissertação. A metodologia aqui descrita serve para que se reduza o tempo despendido na formulação e resolução do problema, minimizando a ocorrência de erros na elaboração do programa de comando da bancada de simulação.

Assim, utilizando os passos seguidamente descritos, o desenvolvedor do programa de comando, mesmo que não esteja muito à vontade com a ferramenta utilizada, corre menos riscos de cometer erros, ou lapsos, durante a elaboração do programa.

Passo 1 – Interpretação do Enunciado do Problema

O primeiro passo a ser realizado é pretendida a interpretação do enunciado do exercício e da sua representação esquemática de forma a extrair a informação sobre o comportamento do sistema a simular. É aconselhável dividir a descrição do sistema em dois níveis sucessivos e complementares, como referido em [44]:

- O primeiro nível descreve o comportamento imposto pelo comando em relação à parte operativa. As especificações funcionais da parte física permitem ao projetista compreender como funciona o sistema face às situações que poderão surgir. Assim as especificações funcionais têm como objetivo dar a compreender qual o papel do comando a construir.
- Neste primeiro nível não são consideradas as especificações tecnológicas, as características dos atuadores, sensores ou outros componentes do sistema. Não interessa, por exemplo, como se dá um certo deslocamento (se por um cilindro ou motor elétrico). O que importa é em que circunstâncias de comando se efetua esse deslocamento.
- O segundo nível acrescenta às especificidades funcionais as especificidades tecnológicas, características de como o automatismo se insere fisicamente no sistema automatizado. Neste nível há que ter em conta as exatas características dos componentes a utilizar no sistema e as limitações que daí podem surgir. Estas especificidades em complemento das especificações funcionais são as que realmente permitem projetar um automatismo que realmente comanda a parte operativa.

Resumidamente, nesta parte da elaboração do programa de comando deve-se numa primeira fase ter em atenção o funcionamento/comportamento do sistema a automatizar e em seguida ter em conta as suas especificidades tecnológicas. Deve-se então, neste passo, entender clara e precisamente, quais as características do equipamento a realizar. Qual o seu comportamento e posteriormente saber quais os sensores e atuadores presentes e como funcionam (atuadores mono ou biestáveis, entre outras coisas).

Passo 2 – Construção das Tabelas com as Entradas e Saídas do Controlador

Como referido no capítulo 3, durante o processo de simulação, o programa de comando e a parte física da simulação e vão interagir através de variáveis Booleanas e não-Booleanas, que simulam as trocas sinais elétricos entre controlador real e parte física real.

- As entradas do controlador correspondem às saídas dos modelos da parte física, como, por exemplo, sinais enviados pelos sensores do sistema,…;
- As saídas do controlador correspondem às entradas dos modelos da parte física, como por exemplo, a atuação de relés, válvulas, motores,….

Também, no que diz respeito à interação homem-máquina (HMI):

- As entradas do controlador correspondem à atuação de instrumentos, pelo utilizador, como por exemplo botoneiras, pedais,…;
- As saídas do controlador correspondem a sinais que "ajudam" o operador a tomar decisões na utilização do equipamento, como por exemplo sinais luminosos, sinais sonoros,…

Por isso, todos os elementos comuns à parte de comando e à parte física devem ter o mesmo nome e codificação (endereço) quando introduzidas no ambiente de trabalho.

Passo 3 – Elaboração do Grafcet de Comando

Após a definição das entradas e saídas passa-se à elaboração do Grafcet de comando, utilizando os sensores descritos no enunciado, de maneira a que este descreva, formalmente, o comportamento pretendido para o sistema. Para isto foram seguidas as regras apresentadas na norma IEC 60848 [5].

Sabendo isto tem-se que escolher qual o tipo de estrutura a utilizar. Existem variadas maneiras de elaboração de um Grafcet. Este pode ser elaborado um Grafcet em sequência única ou paralela, que descreva todo o comportamento do sistema ou mesmo vários Grafcets parciais que comandem cada um dos modelos físicos do sistema automatizado. Estes últimos são um conjunto de Grafcets sincronizados entre si para comandarem um sistema automatizado mais ou menos complexo.

O Grafcet de sequência única é constituído por um conjunto de etapas que vão sendo ativadas umas atrás das outras. Este é mais aplicado a casos simples. Caso o sistema a automatizar seja mais complexo e possua várias ações distintas simultâneas elaboradas por componentes distintos, é mais adequado empregar um Grafcet para cada componente e, posteriormente, sincronizá-los. Por exemplo o exercício do aspirador presente no Anexo C podia ser resolvido em sequência única porém torna-se mais simples resolvê-lo utilizando Grafcets sincronizados.

O Grafcet de sequências paralelas caracteriza-se pela existência de sequências únicas que são ativadas, de forma simultânea, por uma mesma transição. Depois da ativação das distintas sequências a sua evolução produz-se de forma independente. Este tipo de Grafcet é mais adequado a ser utilizado para a descrição do comportamento de sistemas em que uma recetividade origina várias ações simultâneas, como por exemplo o exercício 10 presente no anexo C.

No caso de ser elaborado um Grafcet para cada componente do sistema automatizado, após a elaboração dos vários Grafcets, é necessário proceder à sincronização de todos eles.

Resumindo, os Grafcets podem ser elaborados utilizando diferentes tipos de estruturas, porém o uso de umas em detrimento de outras pode facilitar ou dificultar a sua construção.

Passo 4 – Conversão do Grafcet de Comando para Linguagem de Programação Ladder

A linguagem *Ladde*r foi a escolhida para a elaboração dos programas para PLC, pelo simples facto de ser a mais utilizada, mundialmente, na programação de autómatos (garantindo uma maior abrangência deste trabalho) e, também, pelo facto de os equipamentos e *softwares* utilizados para a programação estarem preparados para serem programados nesta linguagem. Como o formalismo SFC (IEC 60848) [5] é utilizado na modelização do controlador é necessário proceder à sua modelização algébrica, tal como apresentado em [45], de forma que

as equações obtidas sirvam de base à obtenção dos programas em linguagem Ladder que se pretende implementar.

Nesta secção pretende-se mostrar mais aprofundadamente como converter uma especificação SFC num conjunto de equações algébricas.

Esta conversão poderá compreender três módulos que serão executados sequencialmente [14]:

- Cálculo das condições de transposição das transições;

- Cálculo das variáveis de atividade de etapa;

- Cálculo das ações;

- Cálculo das temporizações e contagens;

- Cálculo de funções auxiliares (por exemplo, flancos ascendente e descendente de variáveis).

Estes dois últimos pontos referidos não são aqui sistematizados (nos parágrafos seguintes) mas são considerados na elaboração dos programas para PLC.

Cálculo das Condições de transposição das transições

Segundo [44]: "*Seja CT(q) (Condição de transposição da transição q) uma variável booleana associada a cada transição do SFC. Uma transição (q) (figura 8) pode ser transposta se estiver validada (todas as etapas precedentes estiverem ativas) e se a recetividade a si associada for verdadeira. Então a CT(q) pode ser formulada da seguinte forma*":

$$CT(q) = (\prod_{j=1}^{m} XM_j).R(q) \qquad \textit{(Eq.1)}$$

Onde:

- XMj: Variável Booleana associada à etapa Mj;
- R(q): Recetividade associada à transição (q).

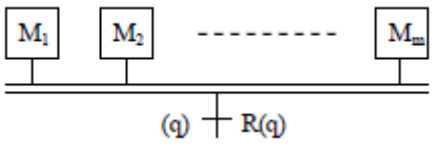


Figura 8 - Recetividade, após sequências simultâneas de acordo com equação 1 [44]

Cálculo das variáveis da atividade de etapa

A formulação geral da atividade de cada etapa pode ser representada de acordo com a equação booleana:

$$X_i(t+1) = \sum_{j=1}^{p} CT(p_j) + X(t).\prod_{k=1}^{n} \overline{CT(n_k)} \qquad \textit{(Eq. 2)}$$

Onde:

- CT(pj): Condição de transposição da transição (Pj)
- p: transição anterior à etapa i
- n: transição posterior à etapa i

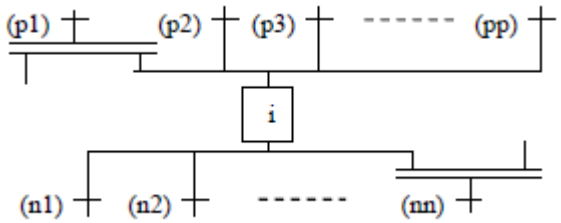


Figura 9 - Etapa de acordo com a equação 2 [44]

Cálculo das ações

A formulação das ações são associadas ao cálculo etapas. As ações estão associadas a determinadas etapas e quando a etapa i se encontra ativa assim também o está a ação associada o está. Se por exemplo a etapa i (figura 9) tivesse associada a si a ação C+, a sua tradução algébrica seria descrita por: *C+ = Xi.*

Passo 5 – Escrita do programa no software de programação

Após concluídos os passos anteriormente descritos, procede-se à escrita das equações em *Ladder* no programa no *software* de programação. Neste trabalho é utilizado o CX-Programmer [11], mas poderia ser qualquer outro, de qualquer outro fabricante. As plataformas quando fornecidas já se encontram com os modelos da parte física escritos no

CX-Programmer. Esta parte deste trabalho mais abrangente foi desenvolvida, em paralelo, no contexto de outra dissertação de mestrado [38].

Optou-se - por simplicidade e comodidade - por colocar os modelos da parte física e do controlador no mesmo ficheiro, mas poder-se-ia optar por outras arquiteturas: por exemplo, o programa do controlador poderia estar num PLC e o programa correspondente aos modelos da parte física poderia estar noutro PLC; neste caso, os PLCs poderiam estar ligados em "malha fechada" de forma que as entradas de um correspondessem às saídas do outro e vice-versa. Na figura seguinte é apresentado um exemplo ilustrativo.

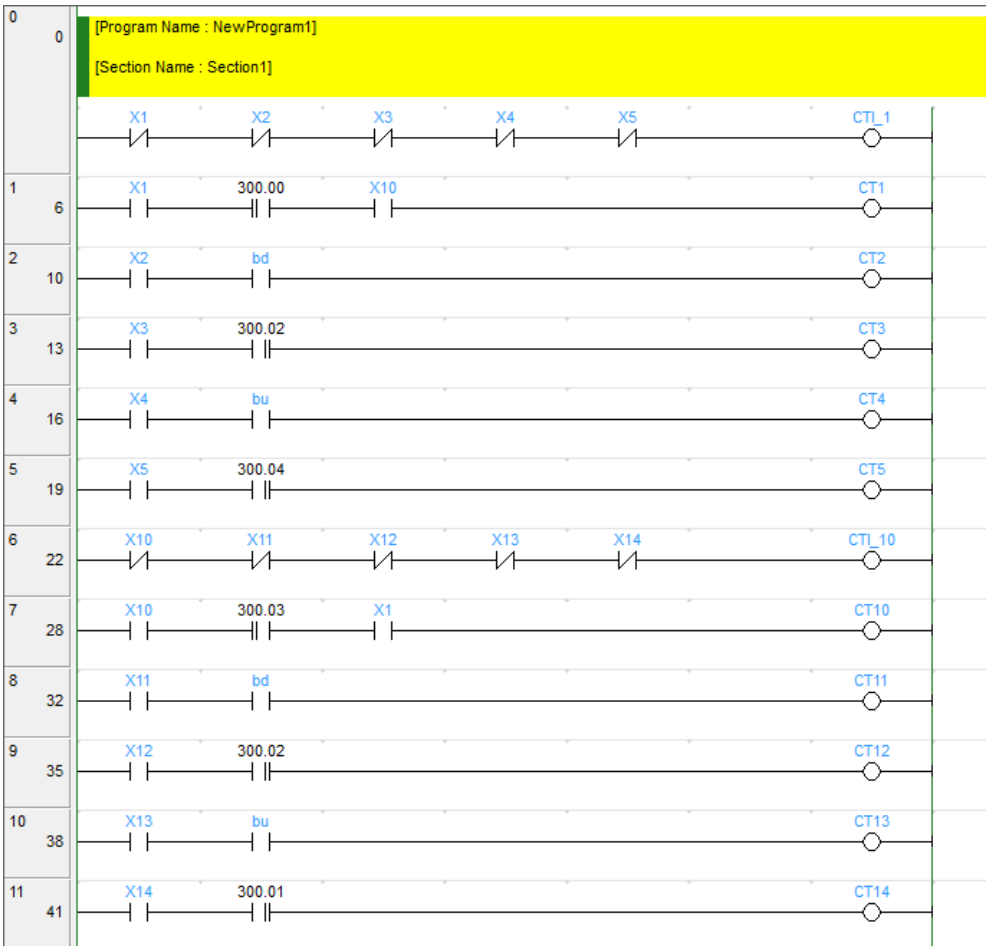


Figura 10 – Exemplo ilustrativo da escrita do Programa de Comando no CX-Programmer.

Passo 6 – Caso a Simulação não Funcione Devidamente

Caso as simulações não funcionem devidamente pode dever-se a vários fatores, entre eles:

- Erros (na estrutura ou na interpretação) na elaboração do Grafcet;
- Erros na conversão para linguagem de programação *Ladder* e escrita das equações no CX-Programmer;
- Erros no modelo da parte física do sistema.

Dado isto, adotam-se os dois passos seguintes, sequencialmente:

1) Verificação de erros na elaboração do Grafcet de comando; Neste primeiro passo de correção verifica-se se o Grafcet respeita as 5 regras da norma IEC 60848 [5] e se esta não contém erros semânticos e de sintaxe;
2) Análise das equações *Ladder* e do CX-Programmer. Neste passo verifica-se se as equações *Ladder* foram traduzidas corretamente a partir da devida conversão do Grafcet para linguagem de programação e também se as equações estão igualmente bem escritas no CX-Programmer ou se houve algum erro de escrita na passagem para este programa. Convém também verificar se os canais das entradas e saídas estão bem definidos.
3) Análise de erros no modelo da parte física do sistema.

4.3. Aspetos Salientados no capítulo 4

Neste capítulo foi evidenciada a metodologia utilizada para a elaboração desta dissertação. Foi apresentado o ambiente de trabalho utilizado e é sugerida uma metodologia de 6 passos para a elaboração do programa de comando das plataformas de simulação.

Esta metodologia de elaboração do programa de comando tem como objetivo a poupança de tempo e trabalho por parte dos utilizadores das plataformas de simulação. Entre os passos sugeridos estão passos que vão desde como interpretar os enunciados dos problemas propostos atá à tradução algébrica do Grafcet para linguagem de programação *Ladder*, sua implementação no ambiente de trabalho e uma sequência de resolução de erros se o programa não funcionar devidamente.

Capítulo 5
CASOS DE ESTUDO

5. Casos de estudo

Como dito no capítulo anterior, neste trabalho foram desenvolvidas 14 plataformas de simulação, correspondentes a 14 casos de estudo. Neste capítulo estão apresentados apenas 3 das 14 plataformas descritas. Estas foram escolhidos conforme o seu grau de complexidade de elaboração do Grafcet de comando, o primeiro caso apresentado trata-se de uma implementação bastante simples, o segundo um caso de dificuldade mediana e o terceiro caso, um caso de difícil elaboração do Grafcet. Todas as restantes encontram-se no Anexo C.

Nestes três exercícios, com diferente grau de dificuldade, estão descritos o enunciado de cada problema, a descrição das variáveis de recetividades e ações, a especificação de comando (ou seja o Grafcet de comando resolvido) e a conversão da especificação de comando para linguagem de programação *Ladder*.

Na especificação de comando utilizou-se a abordagem descrita no subcapítulo 4.2.

5.1. Elaboração da Plataforma de Simulação

A simulação baseia-se nos dois ficheiros. Escritos no CX-Programmer e CX-Designer. No CX-Programmer encontra-se a linguagem de programação (*Ladder*) contendo o programa correspondente à modelização tanto da parte de comando como da parte física. O CX-Designer permite a visualização do estado em que o sistema automatizado se encontra possibilitando o acompanhamento da simulação do programa

Primeiro, o CX-Programmer tem o objetivo de simular o funcionamento da parte operativa e de comando do sistema. A parte operativa encontra-se já implementada no ficheiro do CX-Programmer, já a parte de comando, terá que ser elaborada pelo utilizador da plataforma e implementada no ficheiro por este.

Resumidamente, o utilizador, dependendo da sequência de funcionamento que pretenda que o sistema efetue, terá que desenvolver utilizando da linguagem *Ladde*r o comando do sistema a automatizar, e colocá-la no ficheiro ‘’Exercício X’’ do CX-Programmer. Em seguida pode testar se a sequência foi elaborada corretamente, através do ficheiro ‘’Exercício” do CX-Designer.

Estes ficheiros que já incluem a parte operativa encontram-se na pasta "Exercícios_Bancadas de Simulação" presente no CD disponibilizado com este trabalho.

Para comprovar, se a solução apresentada é idêntica à verificada neste trabalho, pode ser consultada a pasta "soluções_simulação"

5.1.1. Utilização do CX-One na Resolução dos Problemas

- CX-Programmer

Como visto anteriormente após elaborado o Grafcet de comando do sistema automatizado e da sua tradução para linguagem *Ladder*, para isto apenas temos que abrir o ficheiro "Exercicio X" do CX-Programmer e colocar as equações no inicio deste ficheiro, antes das equações (dos modelos físicos) que já lá se encontram (figura 11).

Pelo facto das simulações da parte de comando e operativa estarem simultaneamente em contacto, estas necessitam de possuir o mesmo endereço dos pontos com a mesma função. Assim, o programa colocado no ficheiro terá que respeitar estas condições de endereço apresentadas no capítulo seguinte, após o enunciado proposto por cada exercício.

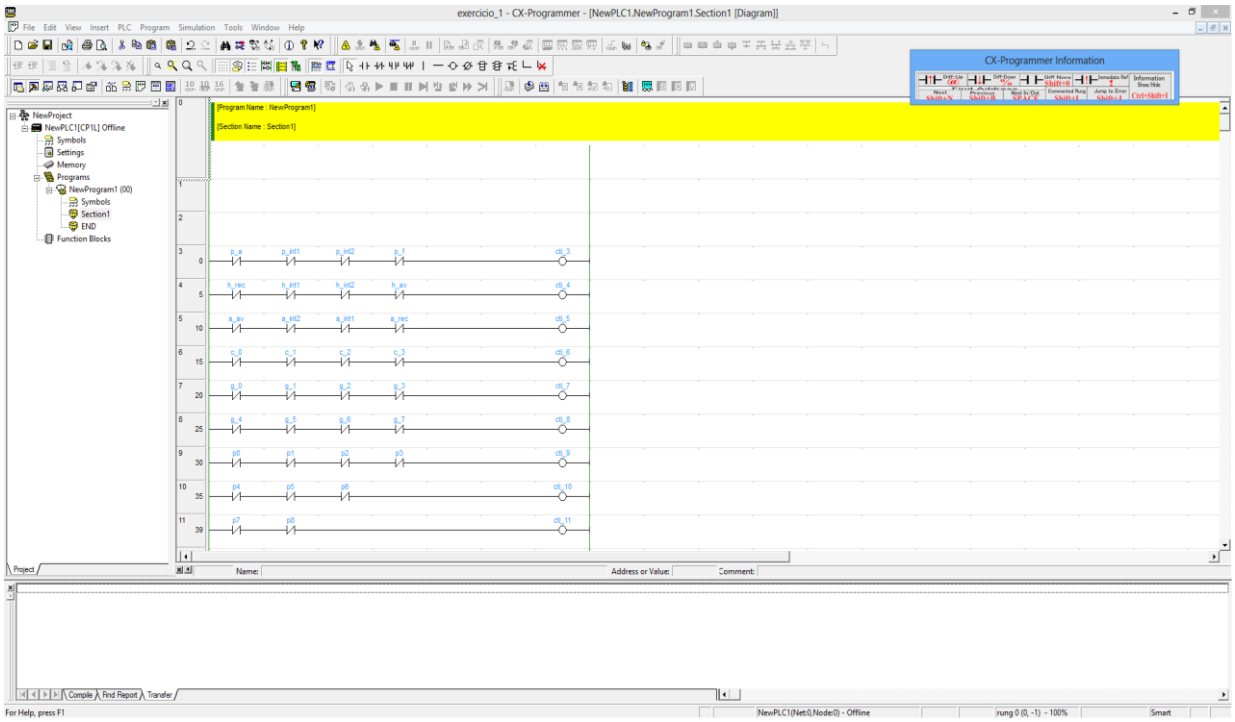


Figura 11 - Resolução dos Problemas Utilizando o CX-Programmer.

- CX-Designer

Por fim, para verificar se a sequência está a funcionar corretamente, abrimos o ficheiro ‘’Exercício X” do CX-Designer (figura 12).

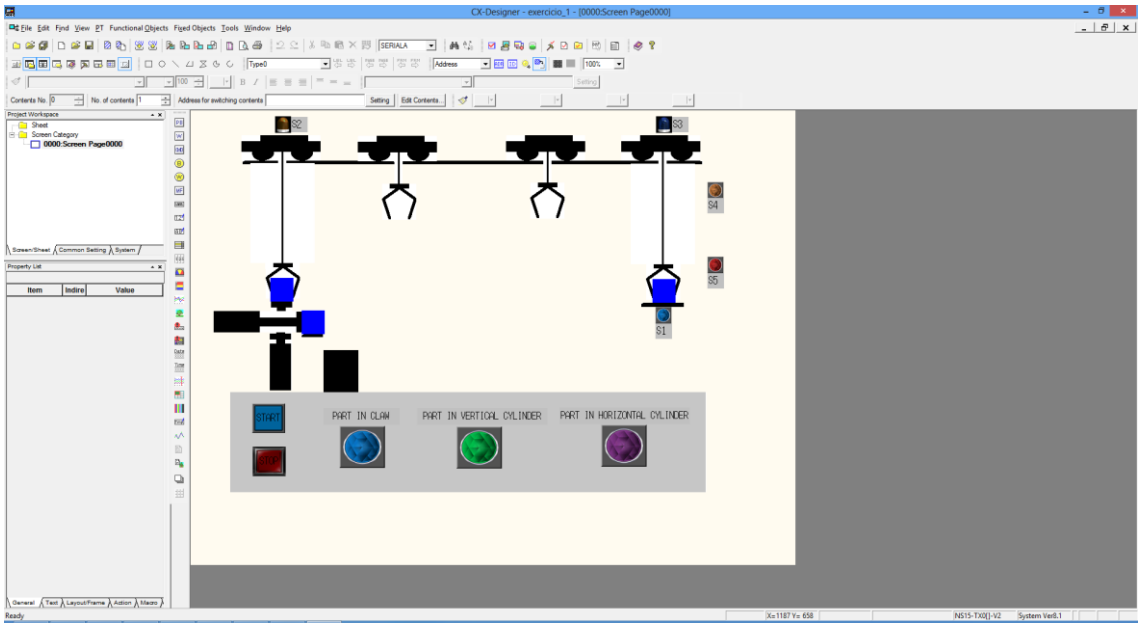


Figura 12 - Resolução dos Problemas Utilizando o CX-Designer.

Em seguida testamos o programa, através de “Tools’’ e “Test’’.

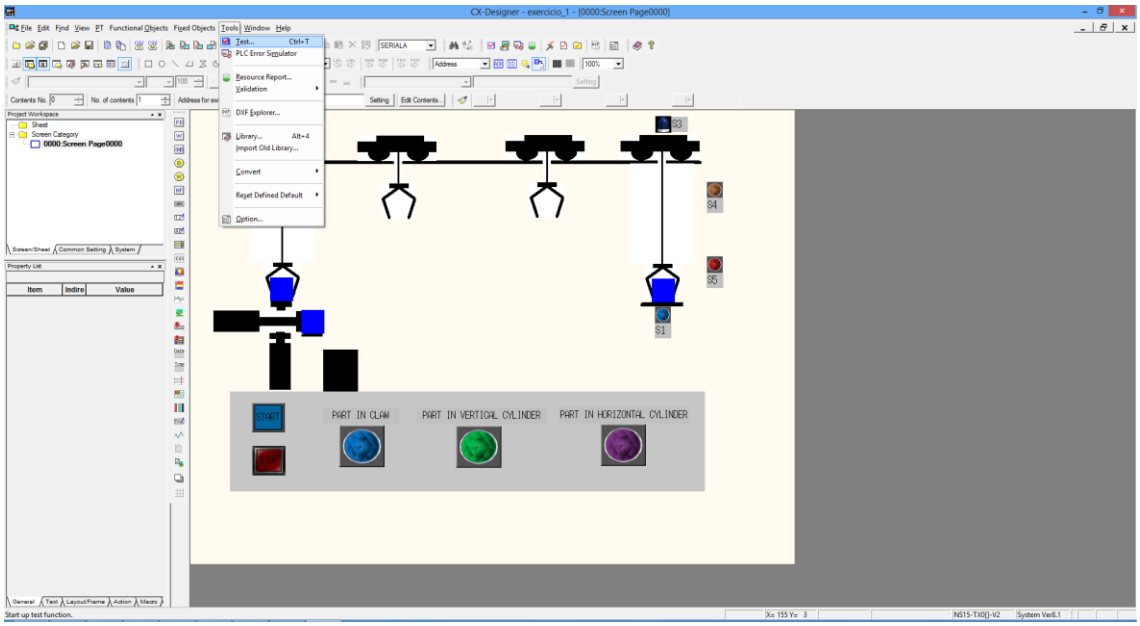


Figura 13 - Resolução dos Problemas Utilizando o CX-Designer.

Em seguida, clicamos na opção ‘’No’’, para não alterar a estrutura de simulação.

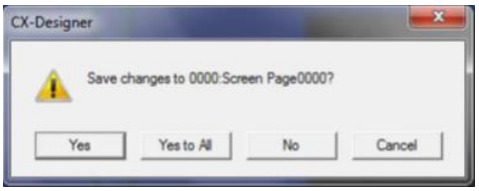


Figura 14 - Resolução dos Problemas Utilizando o CX-Designer.

Por fim clicamos em ‘’Start’’ e o ficheiro de teste abre.

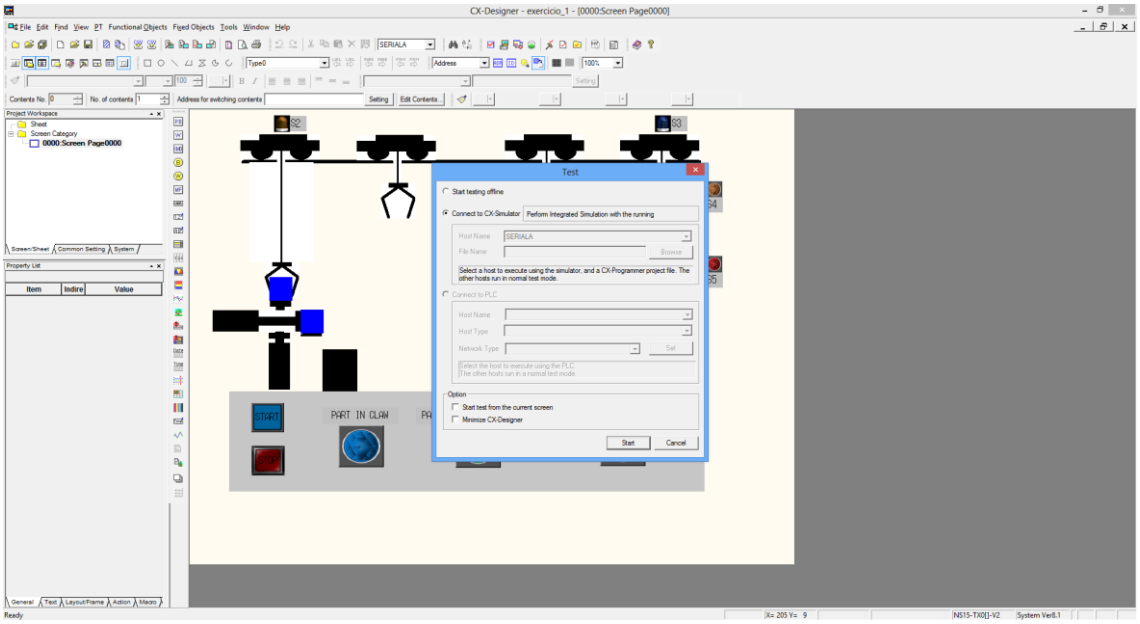


Figura 15 - Resolução dos Problemas Utilizando o CX-Designer.

Com o ficheiro de teste aberto, este pode ser iniciado através da ativação dos botões (Start) e desativado (Stop), conforme o problema proposto.

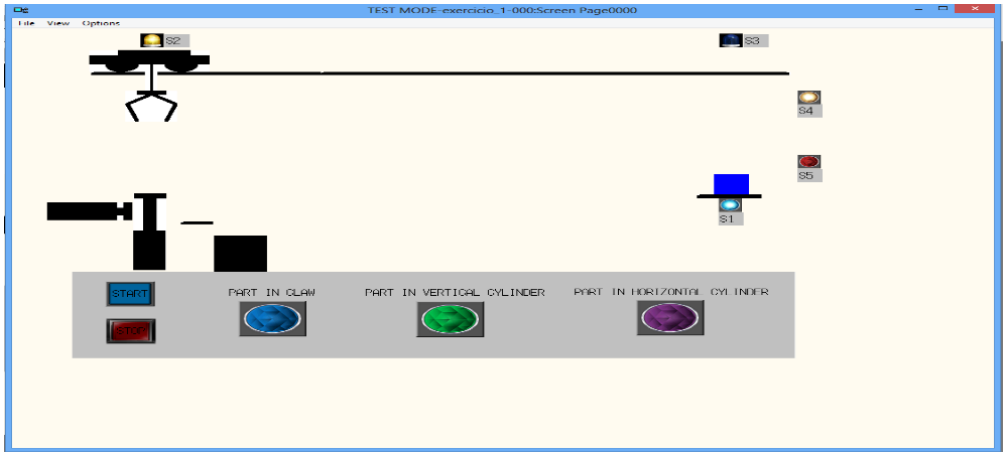


Figura 16 - Resolução dos Problemas Utilizando o CX-Designer.

5.2. Exercício 5 -Passagem de Nível

Pretende-se desenvolver o programa de comando de uma barreira automatizada para uma passagem de nível. Esta move-se para baixo (MD) sempre que o comboio se dirige à passagem de nível e para cima (MU) sempre que este se afasta da passagem de nível. O comboio pode vir de ambas as direções.

A linha férrea possui três sensores (tabela 2) que detetam o comboio: sA, sB e sC como apresentado na figura 17. Também existem dois sensores para a deteção da barreira descida (sbd) e da barreira subida (sbu).

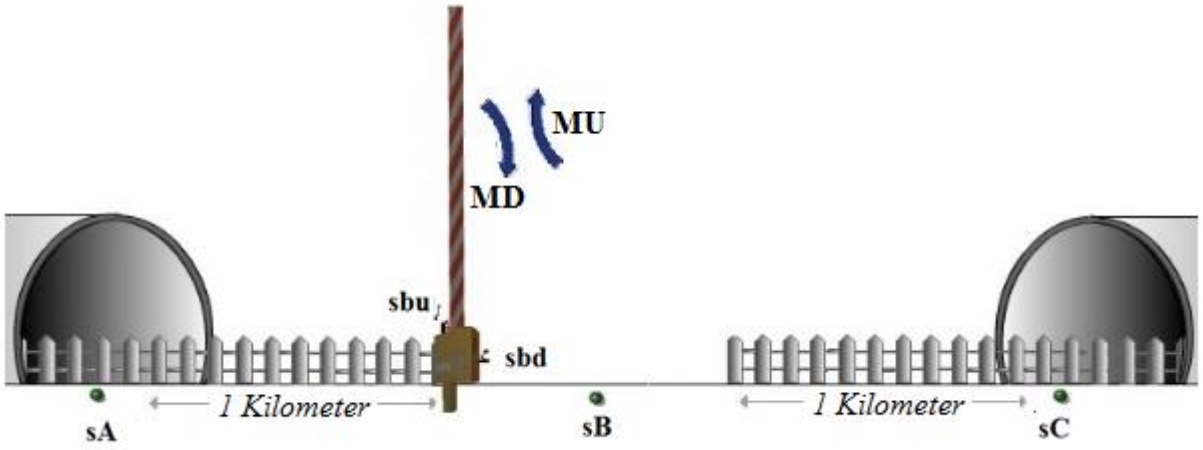


Figura 17 - Esquema da passagem de nível descrito no enunciado do exercício 5.

5.2.1. Modelo do Controlador

A. Variáveis de Entrada e Saídas

O primeiro passo para resolver o exercício é definir as recetividades (inputs) e as ações (outputs) que serão utilizadas no programa de comando. Nas tabelas 2 e 3 estão representadas as descrições, Words e Bits dados às variáveis recetividades e ações respetivamente.

Tabela 2 - Descrição, Words e Bits de todas as Entradas do exercício 5.

Entrada	Descrição	Word & Bit
sA	Comboio em cima do sensor A	0.00
sB	Comboio em cima do sensor B	0.01
sC	Comboio em cima do sensor C	0.02

sbd	Barreira Descida	0.03
sbu	Barreira Subida	0.04

Tabela 3- Descrição, Words e Bits de todas as Saídas do Exercício 5.

Saída	**Descrição**	**Word & Bit**
MU	Barreira move-se para cima	100.00
MD	Barreira move-se para baixo	100.01

B. Especificação de Comando

Para cada exercício, podem ser desenvolvidos várias soluções, todas diferentes mas funcionais.

Assim o utilizador pode desenvolver e testar diferentes programas de comando. O SFC apresentado na figura 18 é apenas uma das possíveis soluções ao problema descrito no exercício 5.

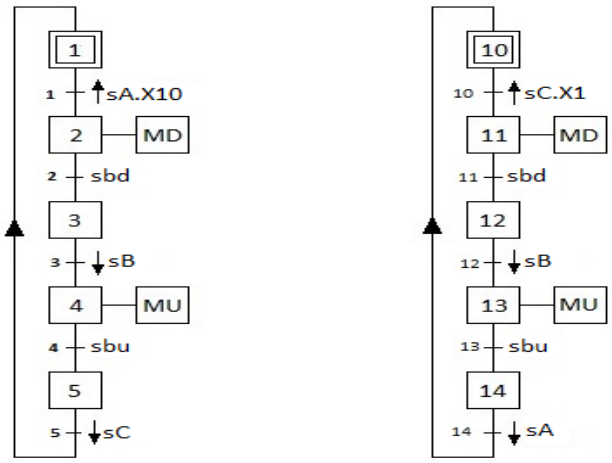


Figura 18 - Representação de uma das possíveis modelizações da parte de comando do exercício 5.

A parte de comando deste exercício possui dois Grafcets, um para cada sentido do comboio. Assim, o Grafcet à esquerda controla a barreira da passagem de nível quando o comboio se movimenta da esquerda para a direita e o Grafcet à direita controla a barreira do comboio quando este se movimenta direita para a esquerda.

C. Conversão da especificação de comando para linguagem *Ladder*

Em função do SFC elaborado para o problema proposto (figura 18), converte-se o Grafcet para linguagem *Ladder*, como enunciado no subcapítulo 4.2.

As condições de transposição das etapas são:

CTi_1 = /X1./X2./X3./X4./X5

CTi_10 = /X10./X11./X12./X13./X14

CT1 = X1.↑sA. X10

CT2 = X2.sbd

CT3 = X3.↓sB

CT4 = X4.sbu

CT5 = X5.↓sC

CT10 = X10.↑sC. X1

CT11 = X11.sbd

CT12= X12.↓sB

CT13 = X13.sbu

CT14 = X14.↓sA

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1+ CT5 + X1./CT1

X2 = CT1 + X2./CT2

X3 = CT2 + X3./CT3

X4 = CT3 + X4./CT4

X5 = CT4 + X5./CT5

X10 = CTi_10 + CT13 + X10./CT10

X11 = CT10 + X11./CT11

X12 = CT11 + X12./CT12

X13 = CT12 + X13./CT13

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

MD = X2 + X11

MU = X4 + X13

5.2.2. Ambiente de Simulação

A modelização da parte de comando do sistema automatizado será simulada com o auxílio do CX-Programmer. As equações de comando, elaboradas na linguagem *Ladder*, serão transpostas para este programa.

A modelização encontra-se representada na figura 19, porém esta não está completa, apenas uma parte da modelização é referida por uma questão de espaço.

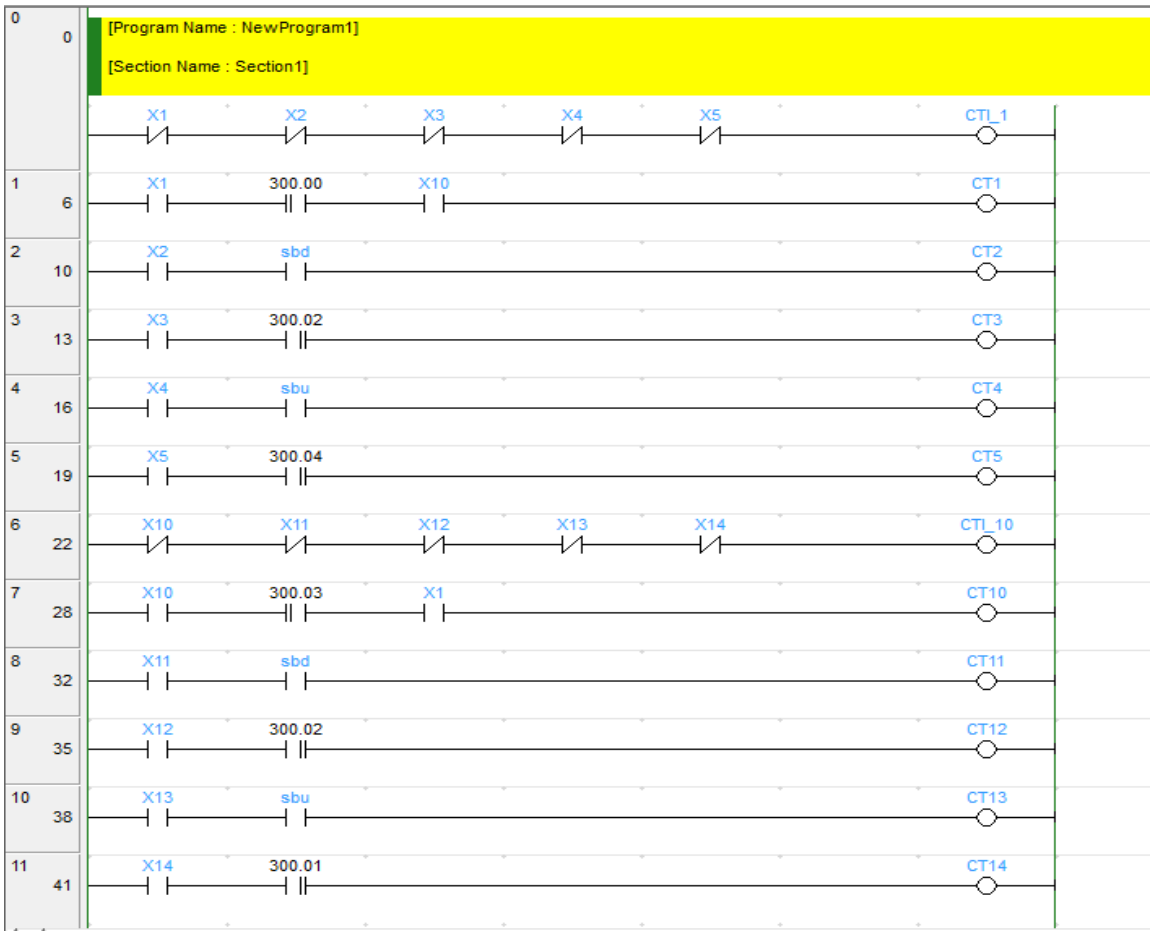


Figura 19 - Representação parcial da modelização da parte de comando do exercício5, no CX-Programmer.

A memória do PLC é simulada através desta modelização, mas para que o comando possa ser executado, é necessário processar as ordens. Para isso utiliza-se o CX-Simulator (figura 20).

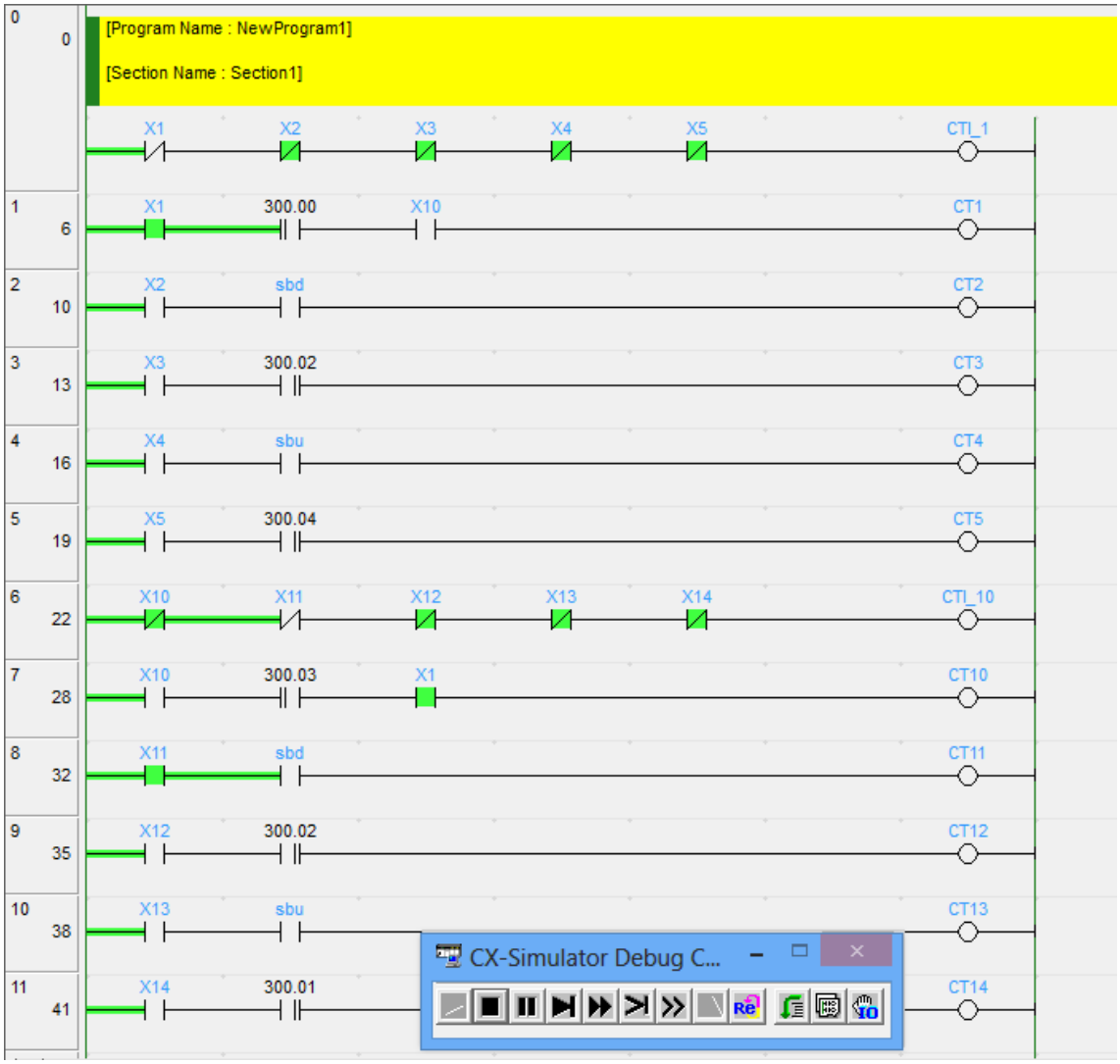


Figura 20 - Representação parcial da modelização da parte de comando do exercício 5 utilizando o CX-Simulator.

Na figura 21 encontra-se a animação da simulação no CX-Designer.

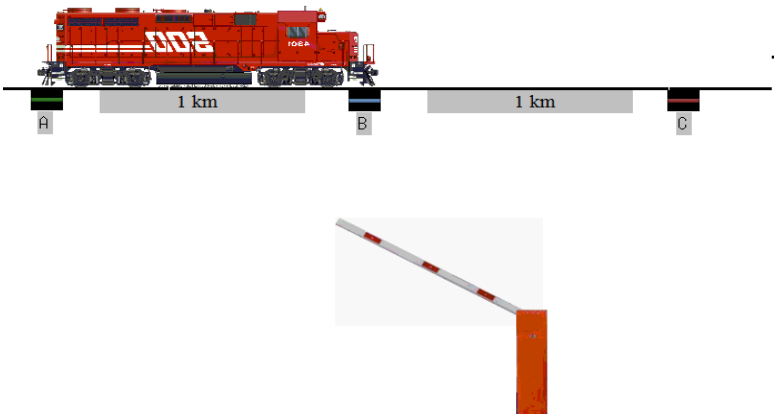


Figura 21 - Animação do Exercício 5 utilizando o CX-Designer.

5.3. Exercício 6 – Aspirador

Tendo em conta o sistema da figura 22, elaborar a especificação do respetivo comando.

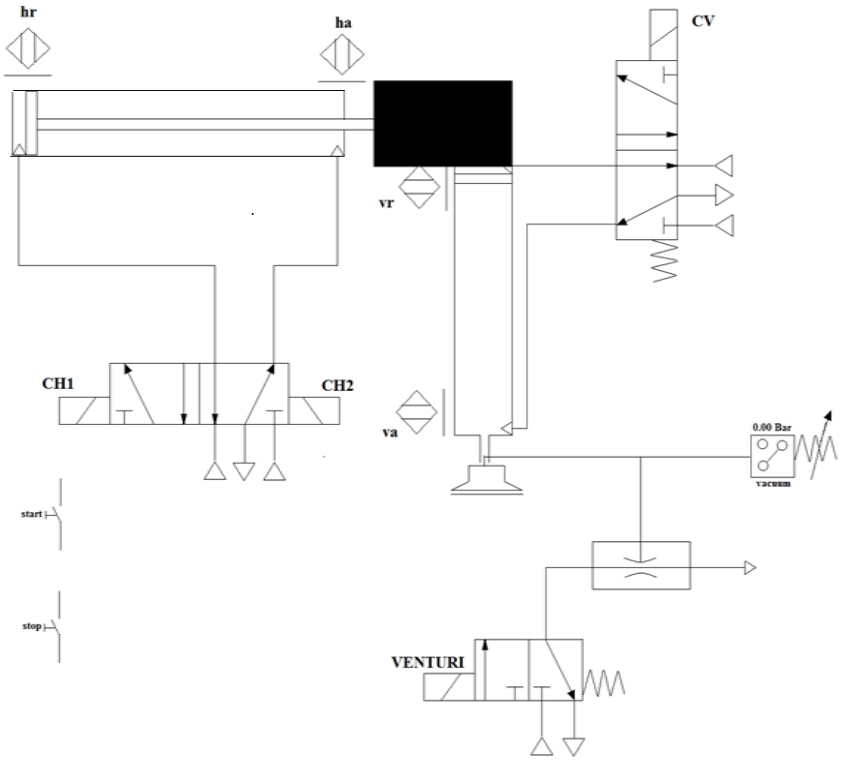


Figura 22 - Sistema Automatizado de Aspiração descrito no enunciado do exercício 6.

Em Grafcet, tendo em conta o segundo percurso:

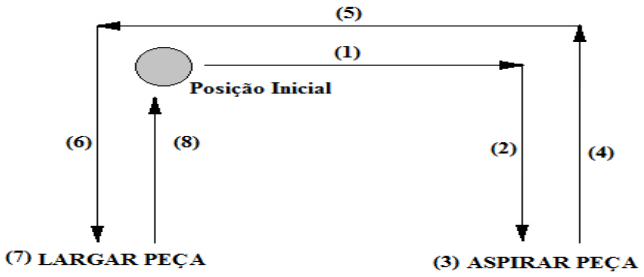


Figura 23 - Percurso percorrido pelo sistema automatizado.

O sistema possui um sensor (tabela 4) que deteta a presença de peça pronta a ser posta no sistema: se1. Também existem quatro sensores de final de curso dos cilindros pneumáticos existentes no sistema: hr, ha, vr, va que representam o cilindro horizontal recuado, avançado e o cilindro vertical recuado e avançado respetivamente. A válvula que comanda o cilindro horizontal é biestável enquanto as que comandam o cilindro vertical e o aspirador são monoestáveis.

As ordens CH1 e CH2 fazem o cilindro avançar e recuar respetivamente; CV ordena o avanço do cilindro vertical; V ordena a aspiração da peça. É de salientar também que para a peça ser aspirada são necessários dois segundos assim como para que a ventosa largue esta.

NOTA: Deve ser considerado um Grafcet para cada cilindro e outro para a aspiração.

5.3.1. Modelo do Controlador

A. Variáveis de Entrada e Saídas

O primeiro passo para resolver o exercício é definir as recetividades (inputs) e as ações (outputs) que serão utilizadas no programa de comando. Nas tabelas 4 e 5 estão representadas as descrições, Words e Bits dados às variáveis recetividades e ações respetivamente.

Tabela 4 - Descrição, Words e Bits de todas as Entradas do Exercício 6.

Entrada	Descrição	Word & Bit
START	Botão de iniciação do sistema.	0.00
hr	Cilindro horizontal recuado	0.01
ha	Cilindro horizontal avançado	0.02
vr	Cilindro vertical recuado	0.03
va	Cilindro vertical avançado	0.04
STOP	Botão de paragem do sistema	0.05
se1	Sensor de deteção de peças prontas para ser aspiradas pelo sistema	0.06

Tabela 5 - Descrição, Words e Bits de todas as Saídas do Exercício 6.

Saídas	Descrição	Word & Bit
CH1	Avanço do Cilindro Horizontal	100.00
CH2	Recuo do Cilindro Horizontal	100.01
CV	Avanço do Cilindro Vertical	100.02
V	Aspiração das Peças	100.03

B. Especificação de Comando

Para cada exercício, pode ser desenvolvidos várias soluções, todas diferentes mas funcionais.

Assim os alunos podem desenvolver e testar diferentes programas de comando. O SFC apresentado na figura 24 é uma das possíveis soluções ao problema descrito no exercício 6.

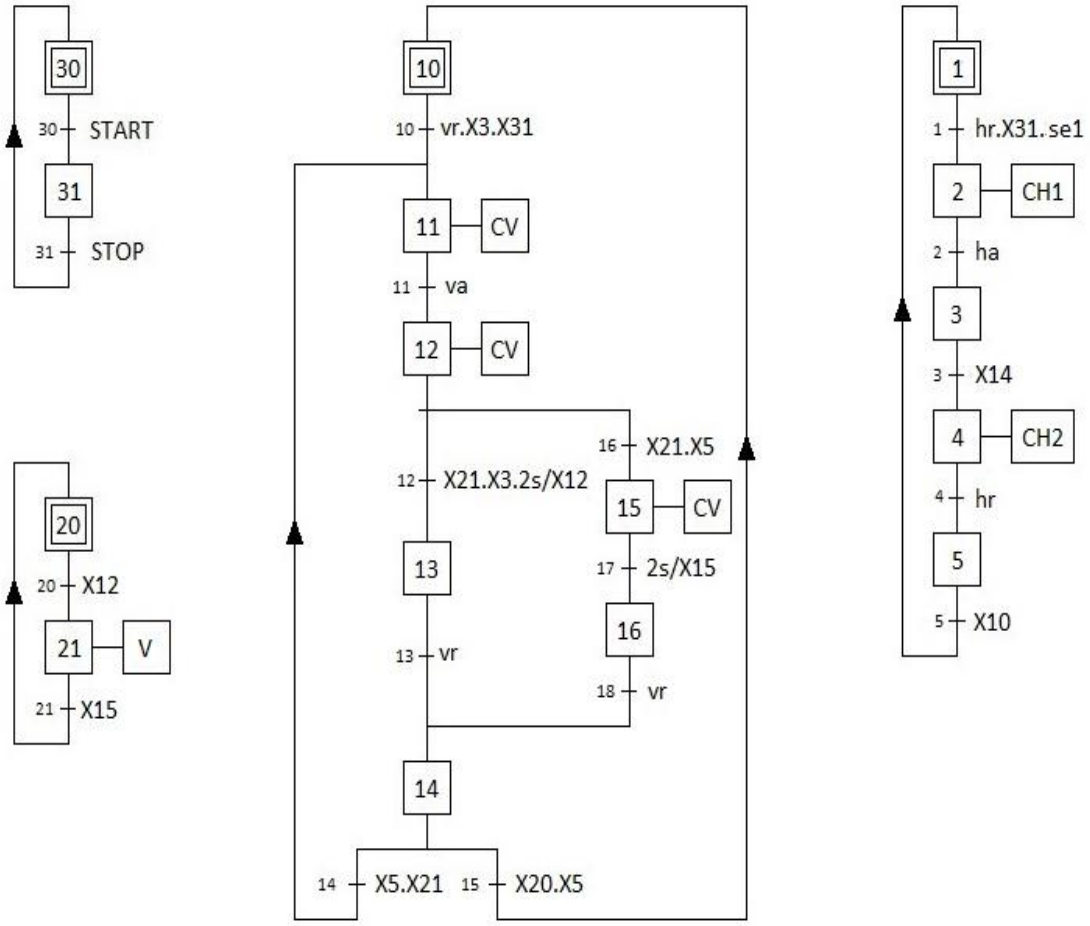


Figura 24 - Representação de uma das possíveis modelizações da parte de comando do exercício 6.

A parte de comando deste exercício possui quatro Grafcets um para cada componente físico do sistema automatizado.

O Grafcet com etapa inicial 10 controla o movimento do cilindro vertical (monoestável); o de etapa inicial 20 controla a aspiração; o de etapa inicial 30 controla o início e paragem do sistema e o de etapa inicial 40 controla o cilindro horizontal (biestável).

C. Conversão da especificação de comando para linguagem *Ladder*

Em função do SFC elaborado para o problema proposto (figura 24), converte-se o Grafcet para linguagem *Ladder*, como enunciado no subcapítulo 4.2.

As condições de transposição das etapas são:

CTi_1 = /X1./X2./X3./X4./X5
CT1 = X1.hr.X31.se1
CT2 = X2.ha
CT3 = X3.X14
CT4 = X4.hr
CT5 = X5.X10
CTi_10 = /X10./X11./X12./X13./X14./X15./X16
CT10 = X10.vr.X3.X31
CT11 = X11.va
CT12= X12.X21.X3.2s/X12
CT13 = X13.vr
CT14 = X14.X5.X21
CT15 = X14.X20.X5
CT16 = X12.X21.X5
CT17 = X15.2s/X15
CT18 = X16.vr
CT20 = X20.X12
CTi_20 = /X20./X21
CT21 = X21.X15
CTi_30 = /X30./X31
CT30 = X30.START
CT31 = X31.STOP

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1+ CT5 + X1./CT1
X2 = CT1 + X2./CT2
X3 = CT2 + X3./CT3
X4 = CT3 + X4./CT4
X5 = CT4 + X5./CT5
X10 = CTi_10 + CT15 + X10./CT10
X11 = CT10 + CT14 + X11./CT11
X12 = CT11 + X12./(CT12 + CT16)
X13 = CT12 + X13./CT13
X14 = CT13 + CT17 + X14./(CT14 + CT15)
X15 = CT16 + X15./CT17
X16 = CT17 + X16./CT18
X20 = CTi_20 + CT21 + X20./CT21
X21 = CT20 + X21./CT21
X30 = CTi_30 + CT31 + X30./CT30
X31 = CT30 + X31./CT31

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

CH1 = X2
CH2 = X4
CV = X11 + X12
V = X15

5.3.2. Ambiente de Simulação

A modelização da parte de comando do sistema automatizado será simulada com o auxílio do CX-Programmer. As equações de comando, elaboradas na linguagem *Ladder*, serão transpostas para este programa (figura 25).

A modelização encontra-se representada na figura 25, porém esta não está completa, apenas uma parte da modelização é referida por uma questão de espaço.

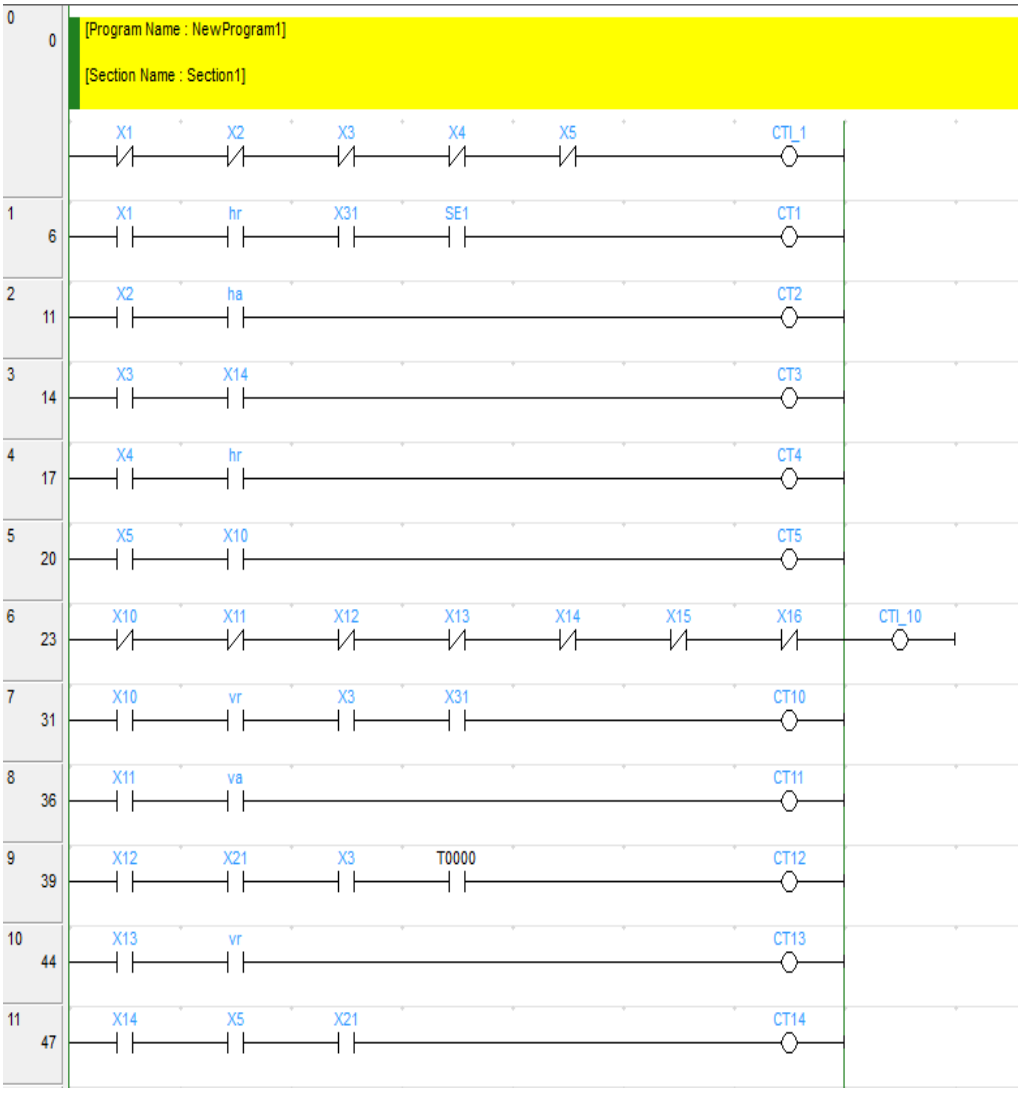


Figura 25 - Representação parcial da modelização da parte de comando do exercício 6, no CX-Programmer.

A memória do PLC é simulada através desta modelização, mas para que o comando possa ser executado, é necessário processar as ordens. Para isso utiliza-se o CX-Simulator (figura 26).

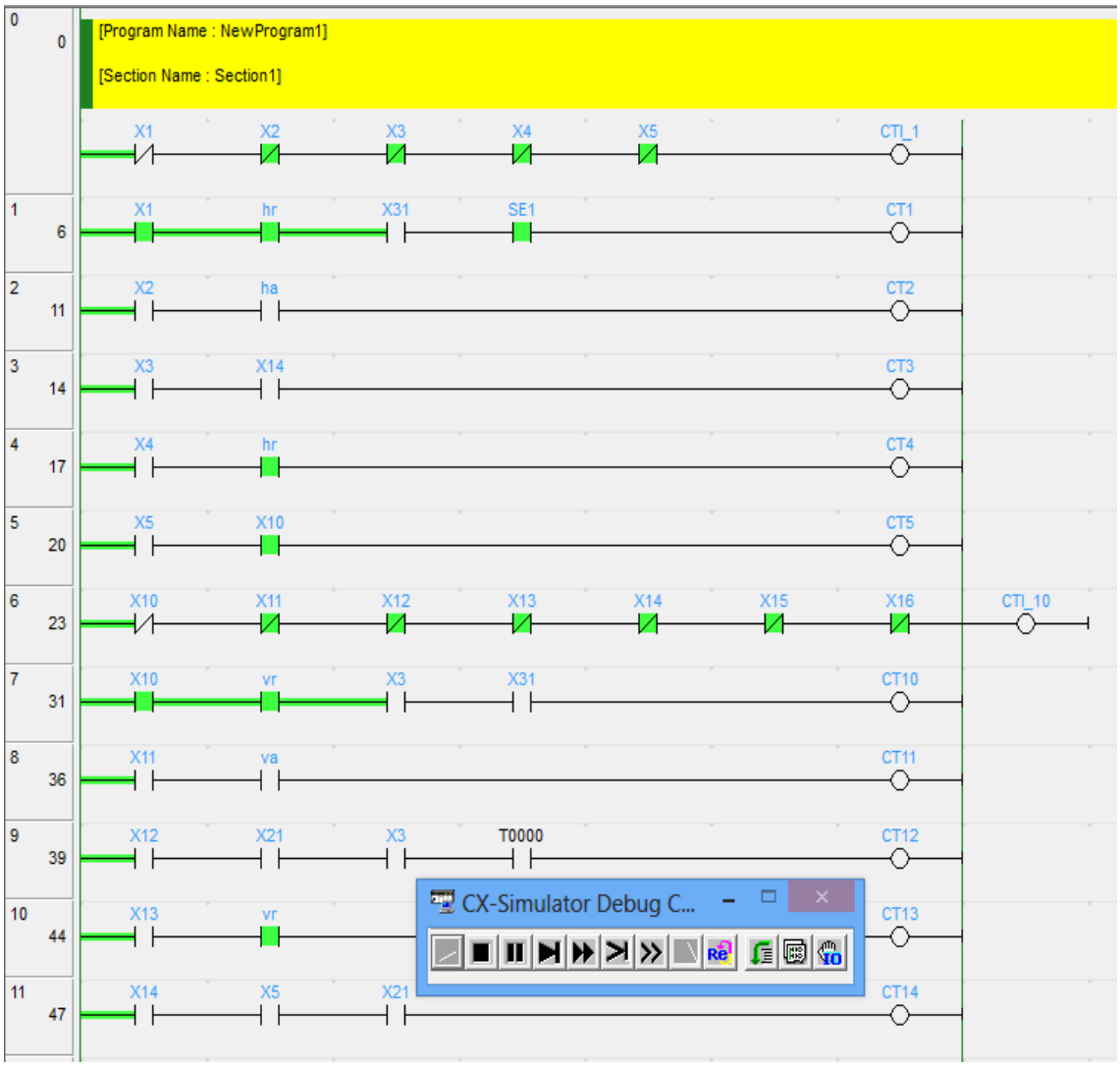


Figura 26 - Representação parcial da modelização da parte de comando do exercício 6,utilizando o CX-Simulator.

Na figura 27 encontra-se a animação da simulação no CX-Designer.

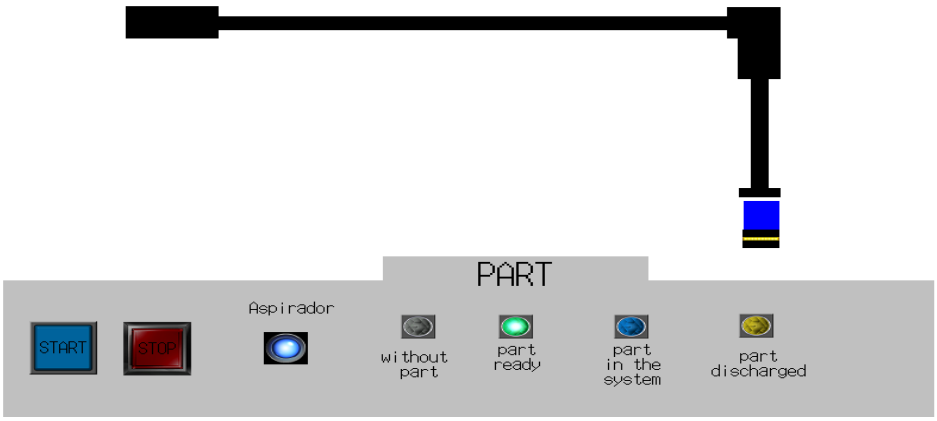


Figura 27 - Animação do Exercício 6 utilizando o CX-Designer.

5.4. Exercício 10 - Estação de furação

Tendo em conta o sistema da figura 28, pretende-se elaborar a especificação do respetivo comando.

O início do ciclo dá-se premindo o botão Start e no início de funcionamento, todos os cilindros devem estar recuados e o motor elétrico deverá estar parado.

Somente é considerado um sensor S1 no sistema para deteção de peças prontas para entrarem no sistema, de resto considera-se um funcionamento normal, onde há sempre peça. Devem ser associados dois fins-de-curso a cada um dos cilindros. Todos estes serão de simples efeito exceto o cilindro F. toas as electroválvulas de comando serão monoestáveis, exceto a electroválvula de comando do cilindro R que será biestável. O sistema possui três postos de trabalho e pretende-se que se realizem as três tarefas em simultânea:

- Carregamento de uma peça;
- Prisão e furação de uma peça;
- Teste e evacuação de uma peça.

O cilindro de rotação permite uma rotação de 120° do prato.

O teste é feito por um cilindro que desce até à sua posição inferior se o furo estiver efetuado. Se após um determinado tempo, o cilindro de teste não descer até à sua posição inferior, significa que o furo não foi corretamente efetuado e então a máquina deve parar até que alguém remova a peça manualmente e rearme a máquina premindo o botão RE_START.

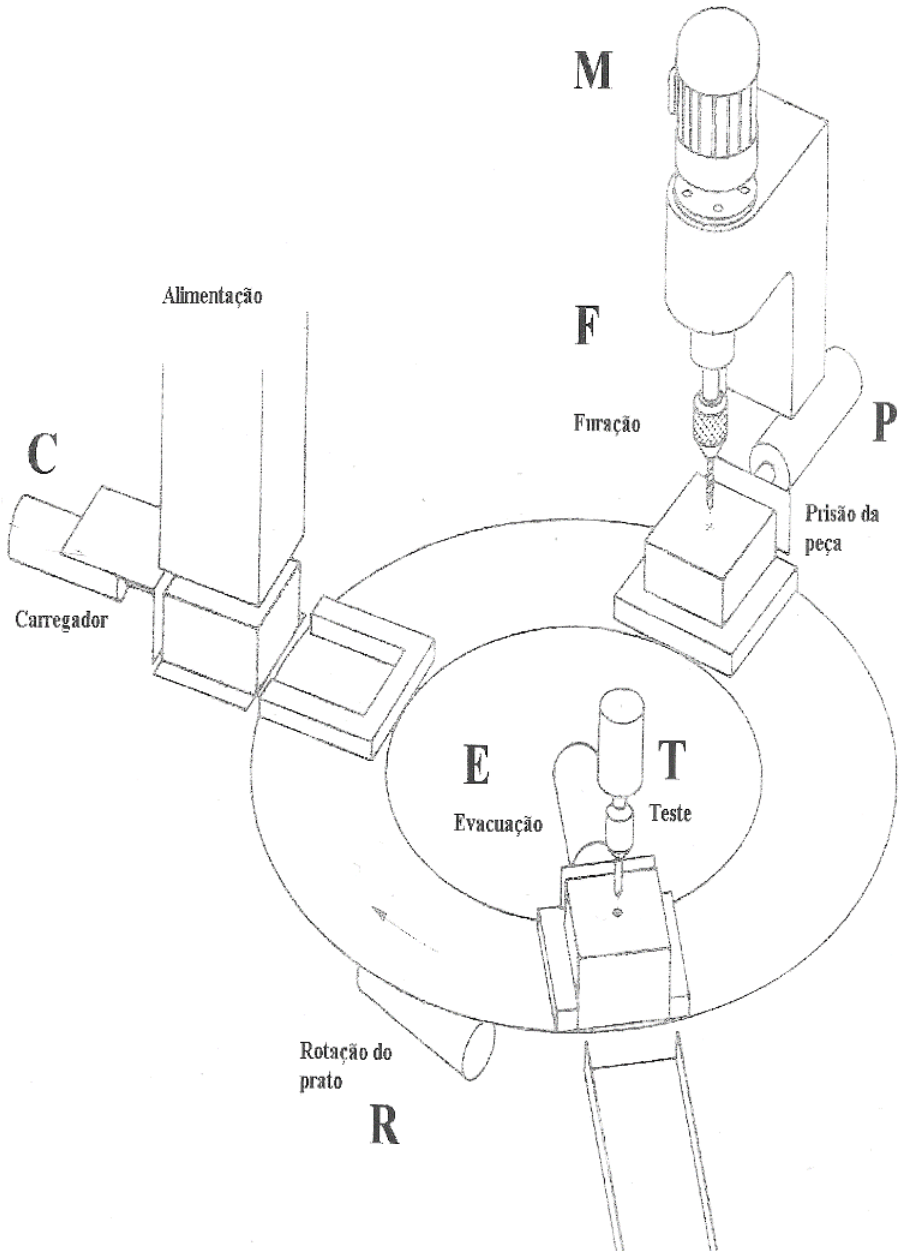


Figura 28 - Estação de Furação Automatizada descrita no enunciado do exercício 10.

5.4.1. Modelo do Controlador

A. Variáveis de Entrada e Saídas

O primeiro passo para resolver o exercício é definir as recetividades (inputs) e as ações (outputs) que serão utilizadas no programa de comando. Nas tabelas 6 e 7 estão representadas as descrições, Words e Bits dados às variáveis recetividades e ações respetivamente.

Tabela 6 - Descrição, Words e Bits de todas as Entradas do Exercício 10.

Entrada	Descrição	Word & Bit
START	Botão de início de funcionamento do sistema	0.00
RE_START	Botão de reinício de funcionamento do sistema	0.01
S_C0	Cilindro C recuado	0.02
S_C1	Cilindro C avançado	0.03
S_P0	Cilindro P recuado	0.04
S_P1	Cilindro P avançado	0.05
S_F0	Cilindro F recuado	0.06
S_F1	Cilindro F avançado	0.07
S_T0	Cilindro T recuado	0.08
S_T1	Cilindro T avançado	0.09
S_E0	Cilindro E recuado	0.10
S_E1	Cilindro E avançado	0.11
S_R1	Cilindro R avançado	1.00
STOP	Botão de paragem do sistema	1.01
S_R0	Cilindro R recuado	1.02
SE1	Sensor de deteção de peças prontas para entrarem no sistema	1.03

Tabela 7 - Descrição, Words e Bits de todas as Saídas do Exercício 10

Saída	Descrição	Word & Bit
EVC	Ordem de avanço do cilindro C	100.00
EVF	Ordem de avanço do cilindro F	100.01
EVP	Ordem de avanço do cilindro C	100.02
M	Ordem de funcionamento do motor	100.03
EVR_AV	Ordem de avanço do cilindro R	100.04
EVR_REC	Ordem de recuo do cilindro R	100.05
EVT	Ordem de avanço do cilindro T	100.06
EVE	Ordem de avanço do cilindro E	100.07

B. Especificação de Comando

Para cada exercício, pode ser desenvolvidos várias soluções, todas diferentes mas funcionais.

Assim os alunos podem desenvolver e testar diferentes programas de comando. O SFC apresentado nas figuras 29 e 30 é uma das possíveis soluções ao problema descrito no exercício 10.

Nesta resolução, comandamos que só se movessem os cilindros que tivessem peça disponível, no primeiro turno apenas existe o carregamento da pela, no segundo existe carregamento e furação e no terceiro em diante já existe carregamento, furação, teste e escoamento da peça. Isto foi conseguido através da adição de contadores à especificação de comando. Também neste comando, promoveu-se o escoamento das peças e trabalho destas enquanto estas eram escoadas do sistema. Assim chegou-se até ao SFC apresentado abaixo.

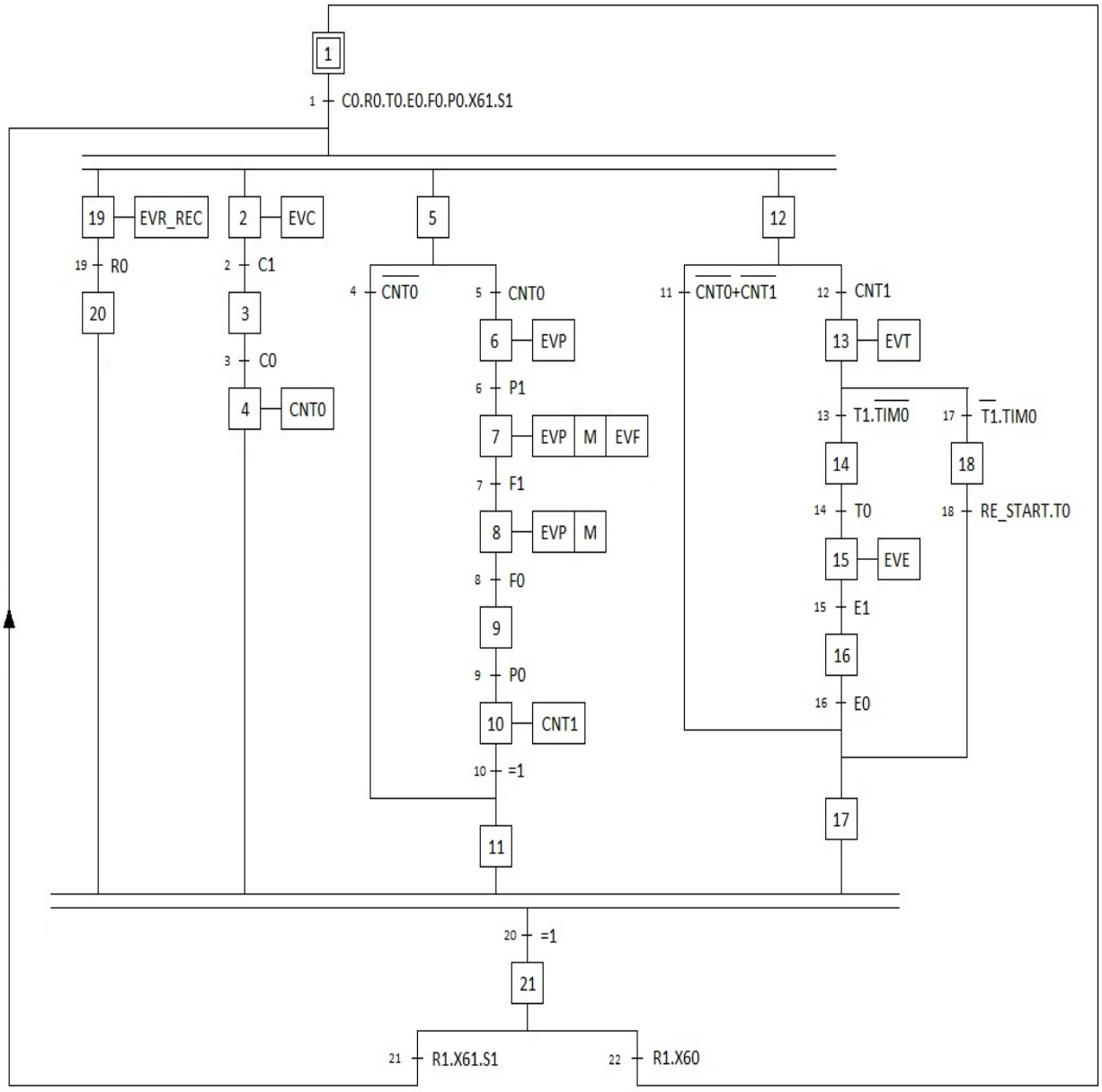


Figura 29 - Representação de uma das possíveis modelizações da parte de comando do exercício 10 (inicio de funcionamento).

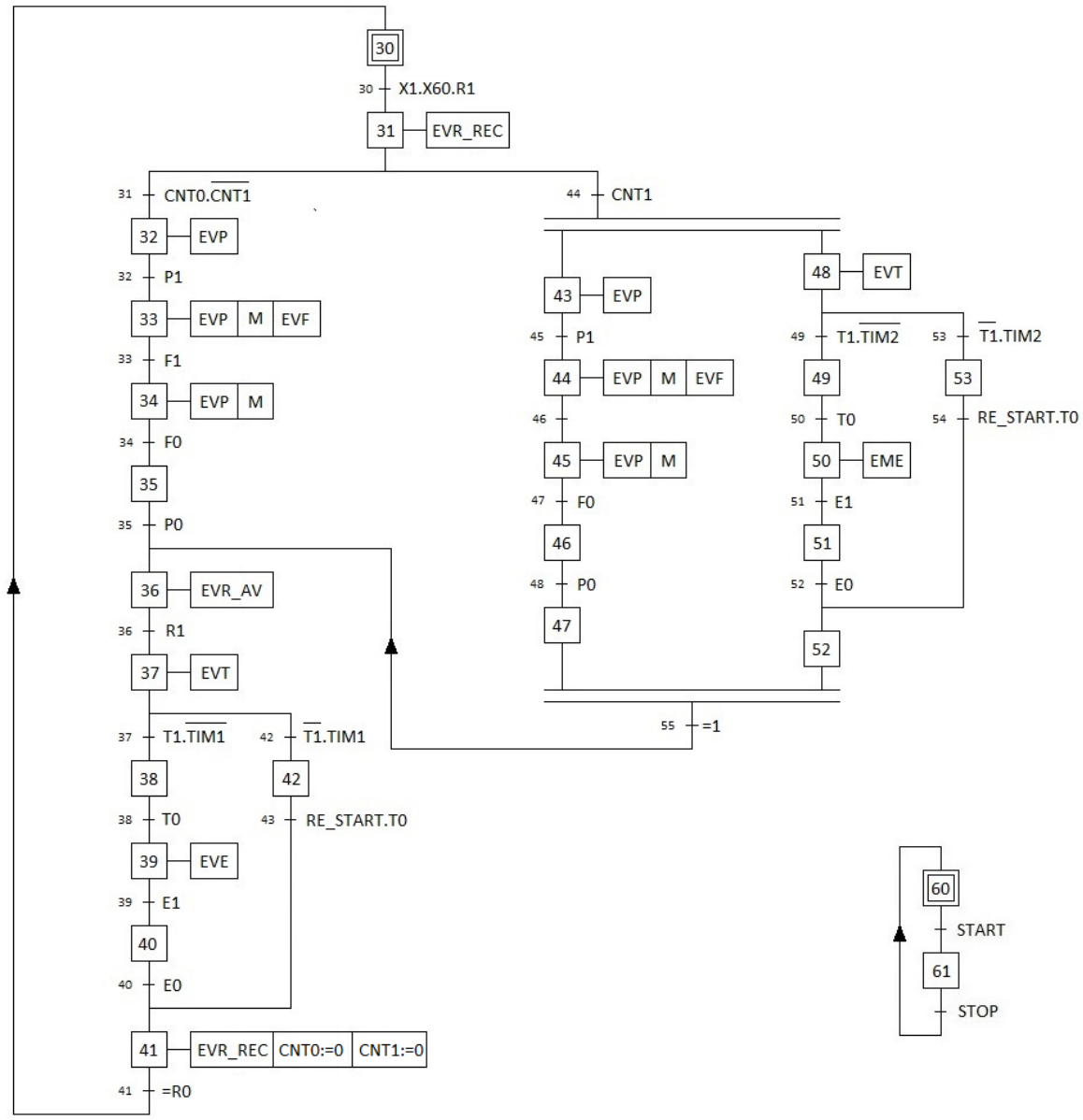


Figura 30 - Representação de uma das possíveis modelizações da parte de comando do exercício 10 (escoamento do produto após paragem).

Como verificado acima a especificação deste comando é constituído por três Grafcets: um de início e paragem do sistema, com etapa inicial 60; um de funcionamento normal da estação de furação, com etapa inicial 1, e outro que ordena o escoamento das peças assim que o sistema é parado, com etapa inicial 30.

O Grafcet de funcionamento normal da estação de furação está pensado de maneira a que no início de funcionamento, apenas funcionem as ferramentas que tiverem peça para ser trabalhada. Desta forma, inicialmente apenas a estação de carregamento, após isso a estação

de carregamento e de furação e depois todas as três estações do sistema e assim adiantes sempre as três, até que seja premido o botão de paragem do sistema e este escoe o produto.

O Grafcet de escoamento também se encontra pensado de maneira a que as peças que se já se encontram no sistema sejas trabalhadas, não permitindo o carregamento de mais nenhuma peça e ordena o funcionamento de apenas as estações que tenham peça.

C. Conversão da especificação de comando para linguagem *Ladder*

Em função do SFC elaborado para o problema proposto (figura 29 e 30), converte-se o Grafcet para linguagem *Ladder*, como enunciado no subcapítulo 4.2.

As condições de transposição das etapas são:

CTi= /X1./X2./X3./X4./X5./X6./X7./X8./X9./X10./X11./X12./X13./X14./X15./X16./X17./X18./X19./X20./X21

CT1 = X1.S_C0.S_R0.S_T0.S_E0.S_F0.S_P0.X61.SE1

CT2 = X2.S_C1

CT3 = X3.S_C0

CT4 = X5./CNT0

CT5 = X5.CNT0

CT6 = X6.S_P1

CT7 =X7.S_F1

CT8 = X8.S_F0

CT9 =X9.S_P0

CT10 = X10

CT11 = X12./CNT0./CNT1

CT12= X12.CNT1

CT13 = X13.S_T1./TIM0

CT14 = X14.S_T0

…

CT54 = X53.RE_START.S_T0

CT55 = X47.X52

CTi_60 = /X60./X61

CT60 = X60.START

CT61 =X61.STOP

Em seguida elabora-se a atividade das etapas:

X1 = CTi + CT22 + X1./CT1

X2 = CT1 + CT21 + X2./CT2

X3 = CT2 + X3./CT3

X4 = CT3 + X4./CT20

X5 = CT1 + CT21 + X5./(CT4 + CT5)

…

X50 = CT50 + X50./CT51

X51 =CT51 + X51./CT52

X52 = CT52 + CT54 + X52./CT55

X53 = CT53 + X53./CT54

X60 = CTi_60 + CT61 + X60./CT60

X61 = CT60 + X61./CT61

EVC = X2

EVF = X7 + X33 + X44

EVP = X6 + X7 + X8 + X32 + X33 + X34 + X43 + X44 + X45

M = X7 + X8 + X33 + X34 + X44 + X45

EVR_AV = X21 + X36

EVR_REC = X19 + X31 + X41

EVT = X13 + X37 + X48

EVE = X15 + X39 + X50

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

5.4.2. Ambiente de Simulação

A modelização da parte de comando do sistema automatizado será simulada com o auxílio do CX-Programmer. As equações de comando, elaboradas na linguagem *Ladder*, serão transpostas para este programa (figura 31).

A modelização encontra-se representada na figura 31, porém esta não está completa, apenas uma parte da modelização é referida por uma questão de espaço.

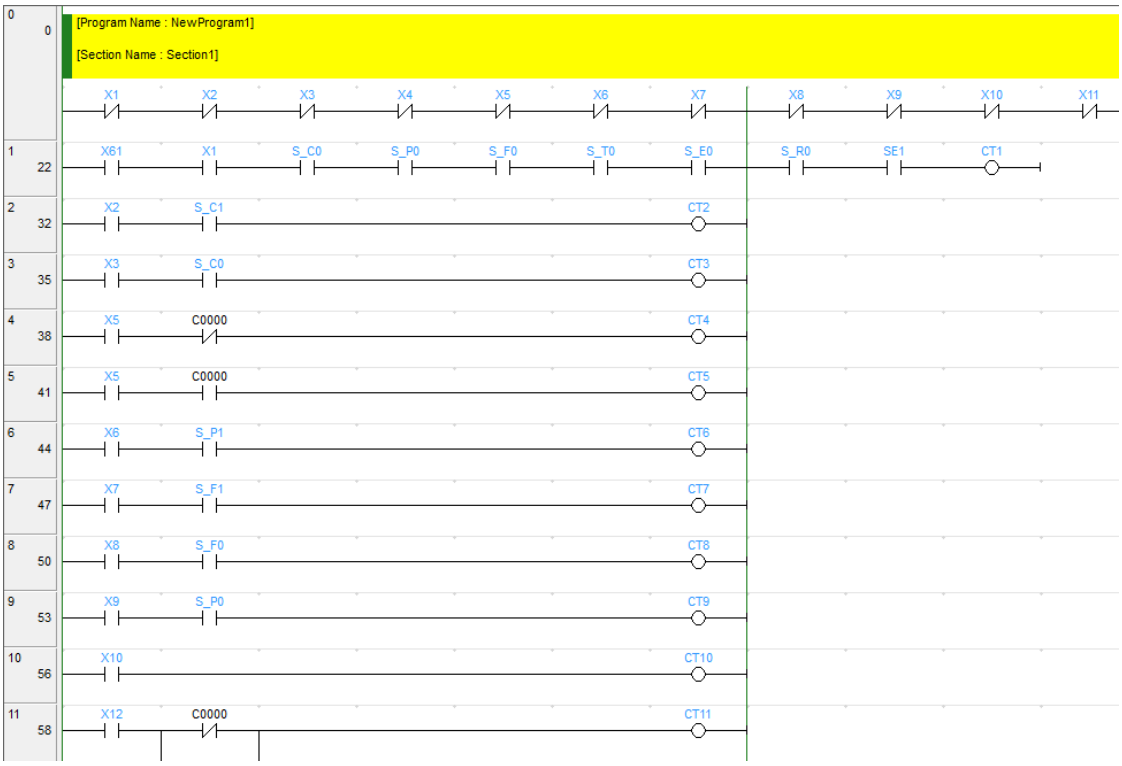


Figura 31 - Representação parcial da modelização da parte de comando do exercício 10 no CX-Programmer.

A memória do PLC é simulada através desta modelização, mas para que o comando possa ser executado, é necessário processar as ordens. Para isso utiliza-se o CX-Simulator (figura 32).

Figura 32 - Representação parcial da modelização da parte de comando do exercício 10 utilizando o CX-Simulator.

Na figura 33 encontra-se a animação da simulação no CX-Designer

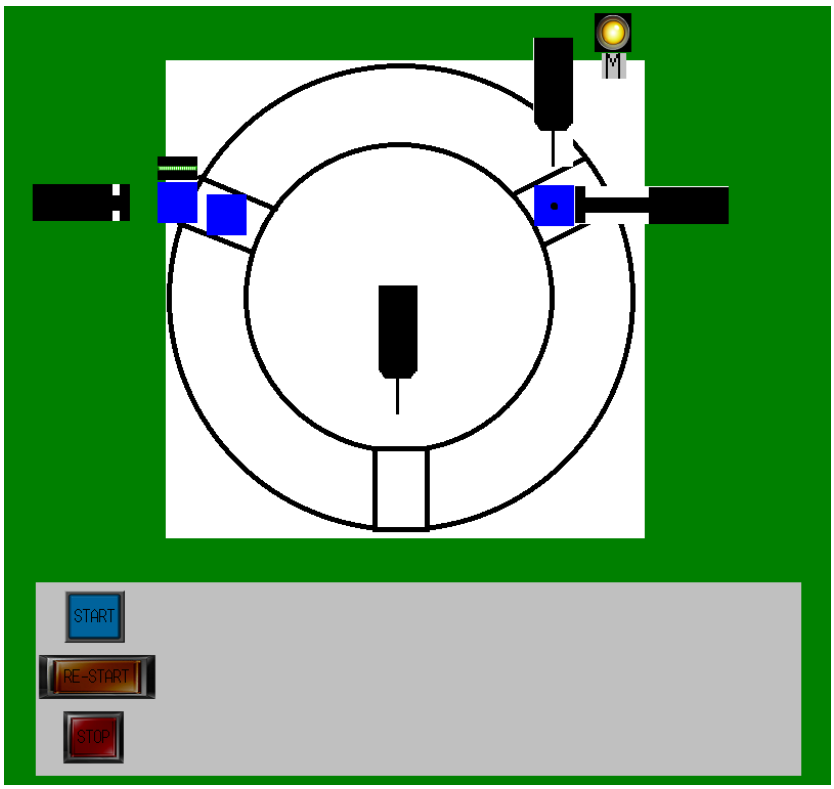


Figura 33 - Animação do Exercício 10 utilizando o CX-Designer.

Capítulo 6
CONCLUSÕES

6. Conclusões

Nos últimos anos com a implementação do Tratado de Bolonha, o contacto de horas entre aluno e professor tem diminuído cada vez mais. Isto implica uma precisa definição dos objetivos e capacidades que o estudante deve adquirir, mas também a reformulação de estratégias e metodologias de ensino. O estudante deve agora ter mais iniciativa e deve ter um papel ativo na sua educação para além do assistir às aulas, escutar o professor e tirar apontamentos. Desta forma os alunos têm que encontrar novas formas de estudo e prática das matérias dadas na aula.

É sabido que os estudantes retêm melhor a informação que estão a aprender se a puderem aplicar na prática. Isto é especialmente importante para os estudantes de engenharia pois estes terão que se confrontar com situação práticas reais durante toda a sua vida profissional. Dado isto laboratórios e lugares de trabalho para fazer exercícios práticos e pôr a teoria das aulas em prática são da máxima importância. Porém ter e manter estes lugares de trabalho é bastante complicado devido ao custo dos equipamentos, a falta de espaço laboratorial e de pessoal qualificado para ajudar e supervisionar os estudantes.

Para colmatar estas debilidades teve-se em conta a simulação em ambientes virtuais e as animações. Desta forma, os estudantes têm a possibilidade de realizar simulações de exercícios práticos quando e onde quiserem, tendo apenas a necessidade de ter um computador pessoal (PC) e o programa correto.

O objetivo principal deste trabalho de dissertação, que consistiu no desenvolvimento de plataformas para simulações virtuais de sistemas automatizados, foi atingido. O ambiente virtual desenvolvido substitui os sistemas físicos reais (bancadas laboratoriais) em diversas funções destacando-se a sua aplicação no ensino de automação. As simulações utilizam o princípio da simulação Model-In-the-Loop, ou seja, a parte de comando e parte física do sistema operativo são simultaneamente modelizadas, sendo apenas necessário um computador e o *software* CX-One.

A utilização do *software* CX-One e utilização de linguagem de programação *Ladder* foram pré-definidas pela simples questão de já haver uma parceria entre a Universidade do Minho e a empresa OMRON para a elaboração de outros projetos académicos. É de notar que qualquer

ferramenta de trabalho semelhante/concorrentes ao CX-One da OMRON poderia ser utilizado uma vez que as linguagens utilizadas encontram-se normalizadas.

Os sistemas automatizados são compostos por duas partes, a parte de comando e a parte física. Assim para elaborar as plataformas de simulação dividiu-se a plataforma em duas partes: parte de comando, descrito nesta dissertação; e parte física descrita num trabalho complementar a este. A parte de comando é a que dita como o sistema tem que funcionar, o que fazer e em que momento, este atribui ordens ao sistema e recebe deste informação para saber em que estado este se encontra. A parte física representa os componentes presentes no sistema automatizado (cilindros, motores, ventosas, sensores, entre outros).

Neste trabalho, tratando-se de simulação MIL, a parte de comando e a parte física é virtual, ou seja os componentes do sistema estão modelizados virtualmente de maneira a que se comportem como sistemas reais. Para cada componente do sistema existe um modelo virtual desse mesmo componente.

No desenvolvimento das plataformas de simulação dos sistemas automatizados foi então feito para cada plataforma proposta, uma parte comando e uma parte física virtual com os modelos de cada componente da plataforma proposta. Os modelos depois de elaborados foram transcritos para o programa CX-One, onde se pode programar através de linguagem de programação *Ladder*, e fazer o ecrã virtual de maneira a que o funcionamento do sistema seja observável.

Foi desenvolvida uma metodologia passo-a-passo para a elaboração do comando de cada plataforma de simulação, esta tem como objetivo reduzir o tempo despendido na formulação e resolução do problema, minimizando a ocorrência de erros na construção do programa de comando da bancada de simulação. Esta tem 6 passos: Interpretação do enunciado do problema; Construção das Tabelas com as Entradas e Saídas do Controlador; Construção do Grafcet de Comando; Conversão do Grafcet de Comando para Linguagem de Programação *Ladder*; Escrita no CX-Programmer; correção da plataforma e verificação de erros caso a simulação não funcione.

Nos casos de estudo, estão presentes as partes de comando de três sistemas automatizados, onde estão definidos os Grafcets de comando. Na simulação, a parte de comando é modelizada tendo em conta o comportamento do enunciado do problema proposto. Já a parte

de física é modelizada em função dos elementos que cada sistema possui, podendo integrar diferentes modelos de comando, como se de um automatismo real se tratasse.

As simulações propostas na dissertação podem, também ser uma ferramenta complementar às aulas de outras Unidades Curriculares que envolvam sistemas a eventos discretos, em cursos de engenharia Mecânica, Eletrónica, Mecatrónica e Biomédica, podendo ser utilizadas por alunos e professores, dentro e fora da sala de aula. Desta forma dispõe-se de uma ferramenta versátil na qual podem ser simulados sistemas concebidos pelo próprio aluno.

As plataformas desenvolvidas serão testadas nas aulas práticas de Unidades Curriculares de sistemas a eventos discretos, na Universidade do Minho.

Em trabalhos futuros, a plataforma será colocada online para download e poder-se-á utilizar a metodologia desenvolvida nesta dissertação, para ampliar a base de dados de exercícios e ser possível modelizar o maior número de sistemas automatizados possível.

REFERÊNCIAS

[1] N. Carvalho, R. Silveira, C. Leão, J. Machado and F. Soares."Platform WALC: design and development of a PLC network", in *Virtual University 10th International Conference*. Bratislava, Slovak Republic.10-11 Dec. 2009.

[2] E. Matias, P. Oliveira, J. Cunha, E. Pires and F. Soares, "E-Grafcet: A multimédia educational teaching tool",*inControlo 2010, 9th Portuguese Conference on Automatic Control*. Coimbra, Portugal. 8-10 Sep. 2010.

[3] E. Hansen,"The role of interactive video technology in higher education: Case study and proposed framework", *Journal Education Technology*, vol 30 (9), pp 13-21, Sep.1990.

[4] G. Carnevali and G. Buttazo, "A Virtual Laboratory Environment for Real-time Experiments", in *Proc. of the 5th IFAC International, Symposium on Intelligent Components and Instruments for Control Applications (SICICA 2003)*.Aveiro, Portugal. 9-11 Jul. 2003, pp. 39-44.

[5] IEC - International Electrotechnical Commission. "Langue de spécification GRAFCET pour diagrammes fonctionnels en sequence." CEI 60848 2 ed, 2000.

[6] IEC - International Electrotechnical Commission. "Programmable Controllers - Part 3." CEI 61 131-3, 1993.

[7] J. Machado, B. Denis, J-J. Lesage. "A generic approach to build plant models for DES verification purposes, " in *Proc. of the Workshop on Discrete Event Systems*,2006 Ann Arbor, USA. DOI:10.1109/WODES.2006.382508

[8] G. Kunz, E. Perondi, J. Machado. "Modeling and simulating the controller behavior of an Automated People Mover using IEC 61850 communication requirements," in *9th IEEE International Conference on Industrial Informatics*, 2011, Lisbon, Portugal. DOI: 10.1109/INDIN.2011.60349472011

[9] J. Machado, E. Seabra, J.C. Campos, F. Soares and C. P. Leão, "Safe controllers design for industrial automation systems". *Computers & Industrial Engineering*, vol. 60 (4), pp. 635‑653,May 2011. DOI:10.1016/j.cie.2010.12.020

[10] P.C. Carneiro. "Desenvolvimento de protótipos virtuais para utilização em simulação Software-In-the-Loop."Dissertação de Mestrado em Engenharia Mecânica, Universidade do Minho, Guimarães, Portugal, 2012.

[11] Página da Internet: Omron (2011), consultado em 10/08/2013, disponível em: http://omron.pt/pt/home

[12] L.F. Gomez. "Redes de Petri Reactivas e Hierárquicas- Integração de formalismos no projecto de sistemas reactivos de tempo-real." Dissertação de Doutoramento em Engenharia Eletrotécnica, Universidade Nova de Lisboa, Lisboa, Portugal, 1997.

[13] P.Laplante. *Real-Time Systems Design and Analysis, an Engineer's Handbook.* New York, New York, U.S.A. Institute of Electrical & Electronics Engineers (IEEE) Press.1992.ISBN 0-7803-0402-0.

[14] J. Machado. “Concepção e realização do Comando Operacional de Sistemas Industrias de Eventos Discretos.” Trabalho de Síntese, Provas de Aptidão Pedagógica e Capacidade Científica, Universidade do Minho, Guimarães, Portugal, 2001.

[15] M.A. Reis. “Reengenharia de um sistema de controladores domóticos utilizando Redes de Petri”, Dissertação de Mestrado em Engenharia Eletrotécnica, Universidade Nova de Lisboa, Lisboa, Portugal, 2011.

[16] D.Harel. “Statecharts: a visual formalism for complex systems. ”*Science of Computer Programming*, vol 8 (3), pp. 231-274, Jun. 1987.

[17] R.E. Shannon. *Systems Simulation: The Art and Science*. Englewood Cliffs, New Jersey: Prentice-Hall, 1975.ISBN 10: 0138818398 / ISBN 13: 9780138818395

[18] F. Schaf. “Arquitetura para ambiente de ensino de controle e automação utilizando experimentos remotos de realidade mista. ”Dissertação de Mestrado em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 2006.

[19] H. Silva. “Simulação com Hardware In the Loop Aplicada a Veículos Submarinos Semi-Autônomos.” Dissertação de Mestrado em Engenharia Mecatrónica e de Sistemas Mecânicos, Universidade de São Paulo, São Paulo, Brasil, 2008.

[20] A.F. Paiva. “Geração Automática de Modelos de Simulação de uma Linha de Produção na Industria Textil.” Dissertação de Mestrado em Engenharia Industrial, Universidade do Minho, Guimarães, Portugal, 2005.

[21] D. Chioran, J. Machado. “Design of a Mechatronic System for Application of Hardware-in-the-loop Simulation Technique,” in*3rd International Conference on Innovations, Recent Trends and Challenges in Mechatronics, Mechanical Engineering and New High-Tech Products Development*. 22-23 Sep. 2011. Bucharest, Romania.

[22] C. Bonivento, M. Cacciari, A. Paoli and M. Sartini. “Rapid prototyping of automated manufacturing systems by software-in-the-loop simulation,” in *Chinese Control and Decision Conference (CCDC)*, 23-25 May 2011, pp 3968–3973. Mianyang, China. DOI: 10.1109/CCDC.2011.5968915

[23] J.S. Keranen T.D. Raty, “Model-based testing of embedded systems in hardware in the loop environment”. *IET Softw*.,vol. 6 (4), pp. 364–376, 2012. DOI: 10.1049/iet-sen.2011.0111

[24] F. Hillier, G.S. Liebermann. *Introduction to operations research*. New York: McGraw-Hill, 1988.

[25] C.D. Pegden et. al. *Introduction to Simulation Using Siman*. New York: McGraw-Hill, Mar. 1991.

[26] R.B. Chase, N.J. Aquilano. *Production and Operations Management*. Homewood, Illinois: Irwin,1989.

[27] T.H. Naylor. *Computer Simulation Tecniques*. New York: John Wiley& Sons Inc.,1966. ISBN 10: 0471630608 / ISBN 13: 9780471630609

[28] G. Gordon. *System Simulation.* Englewood Cliffs, New Jersey: Prentice-Hall, 1978.

[29] D.E. Knuth. *The art of Computer Programming: Seminumerical Algorithms*. vol.1. Reading, Massachusetts: Wesley, 1969.

[30] G.I. Doukidis. "An anthology on the homology of simulation with artificial intelligence". *Journal of the Operational Research Society,* vol. 38 (8), pp. 673-681, 1987.

[31] W. Zhu, S. Pekarek, J. Jatskevish, O. Wasynczuk, D. Delisle, "A Model-in-the-Loop Interface to Emulate Source Dynamics in a Zonal DC Distribution System", IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 20, NO. 2, MARCH 2005, 438 – 445.

[32] S. Demers, P. Gopalakrishnan, L. Kant, "A Generic Solution to Software-in-the-loop", Applied Research, Telcordia Technologies, 331 Newman Springs Road, Red Bank, USA.

[33] O. Meister, N. Frietsch, J. Seibold, and G. F. Trommer, "*Software-in-the-loop simulation for small autonomous VtolUav with teaming capability*," in Institute of Systems Optimization, University of Karlsruhe (TH), Karlsruhe, Alemanha.

[34] J. Dantas. "Software de Simulação Hardware-in-the-Loop para a Simulação do Sistema de Navegação e Controle de Veículos Autônomos Submarinos." Trabalho de Conclusão de Curso, Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2008.

[35] Jim A. Ledin, "Embedded Systems Programming: Hardware-in-the-loop simulation", Fev. 1999, 42 – 60.

[36] M. Montazeri-Gh and M. Nasiri. "Hardware-in-the-loop simulation for testing of electro-hydraulic fuel control unit in a jet engine application". *Simulation*, vol. 89 (2), pp. 225-233, Jan. 2013. doi: 10.1177/0037549712466153

[37] O. Ljungkrantz, K. Akesson, M. Fabian, Y. Chengyin (2010). "Formal Specification and Verification of Industrial Control Logic Components". IEEE Transactions on Automation Science and Engineering. 2010, 7; 3, 538 – 548.

[38] N. Canadas. "Modelação da parte física de sistemas mecatrónicos e estudo da sua influência em Simulação MiL (Model-in-the-loop)." Dissertação de Mestrado em Engenharia Mecatrónica, Universidade do Minho, Guimarães, Portugal, 2013.

[39] Página da Internet: Omron (2011) CX-One Consultado em 16/09/2013, disponível em: http://industrial.omron.pt/pt/products/catalogue/automation_systems/software/configuration/cx-one/default.html

[40] Página da Internet: Omron (2011) CX-Programmer, Consultado em 16/09/2013, disponível em: http://industrial.omron.pt/pt/products/catalogue/automation_systems/software/configuration/cx-one/cx-programmer.html

[41] Página da Internet: Omron (2011) CX-Simulator, Consultado em 16/09/2013, disponível em: http://industrial.omron.pt/pt/products/catalogue/automation_systems/software/configuration/cx-one/cx-simulator.html

[42] Página da Internet: Omron (2011) CX-Designer, Consultado em 16/09/2013, disponível em: http://industrial.omron.pt/pt/products/catalogue/automation_systems/software/programming/cx-one/cx-designer.html

[43] Página da Internet: Omron (2011) CX-Supervisor, Consultado em 16/09/2013, disponível em: http://industrial.omron.pt/pt/products/catalogue/automation_systems/software/runtime/cx-supervisor/default.html

[44] "GRAFCET – Diagrama funcional Para Automatismos e Sequencias" Texto pedagógico, Universidade do Minho, Guimarães, Portugal. Consultado em 06/10/2013 disponível em: http://dei-s1.dei.uminho.pt/lic/AUT/sebenta/GRAFCET_Telemec.pdf

[45] J. Machado. "Da Especificação em Grafcet (IEC 60 848) à Implementação em Ladder (IEC 61 131-3)." Texto pedagógico, Universidade do Minho, Guimarães, Portugal, 2003.

ANEXO A – CRIAÇÃO DE UM PROGRAMA NO CX-PROGRAMMER PARA LIGAR A UM PLC VIRTUAL

Criação de um programa no CX-Programmer para ligar a um PLC Virtual.

Pode-se abrir o CX-Programmer através do menu START ou no ambiente de trabalho.

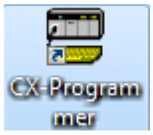


Figura 34 - Atalho do CX-Programmer.

Em seguida, o *software* abre-se, aparecendo aparecendo uma pequena janela:

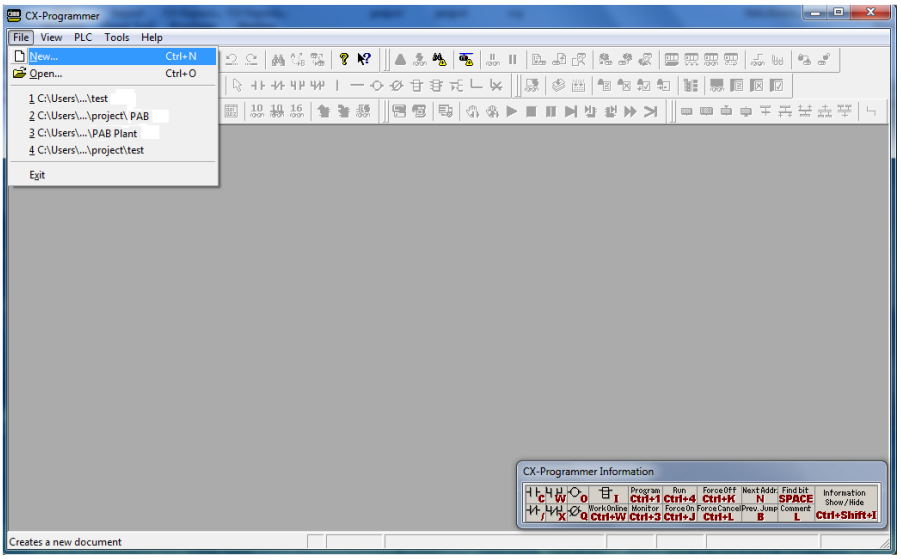


Figura 35 - Pasta "File" do CX-Programmer.

Em primeiro lugar, escolhe-se o nome do servidor do PLC, neste caso, como se pretende utilizar um PLC virtual, escolhe-se o servidor CP1L.

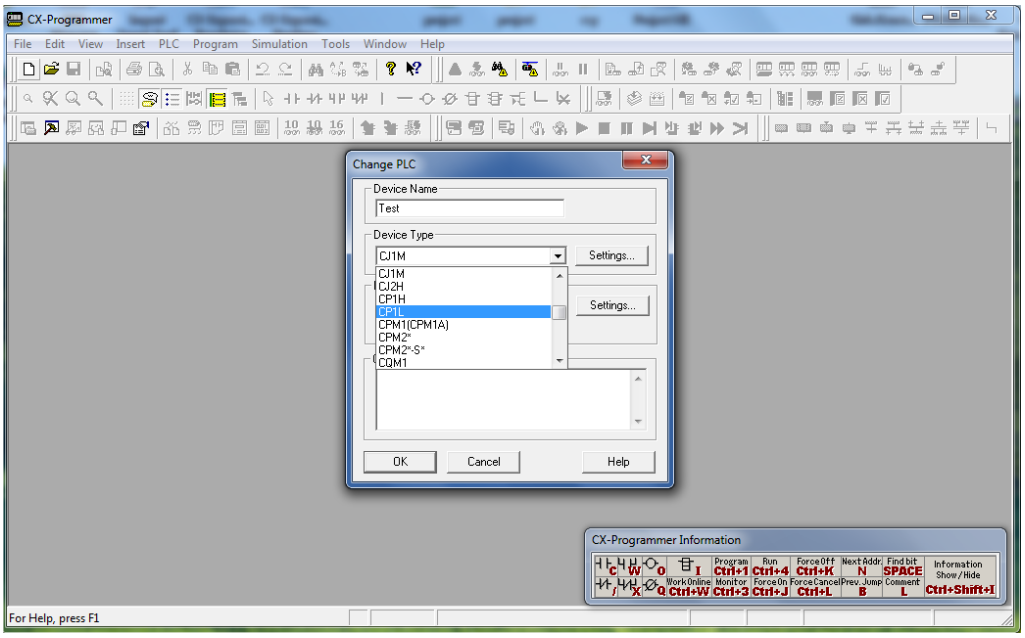


Figura 36–Escolha do Servidor do PLC na Janela "Change PLC" do CX-Programmer.

E na ligação de serviço escolhe-se USB.

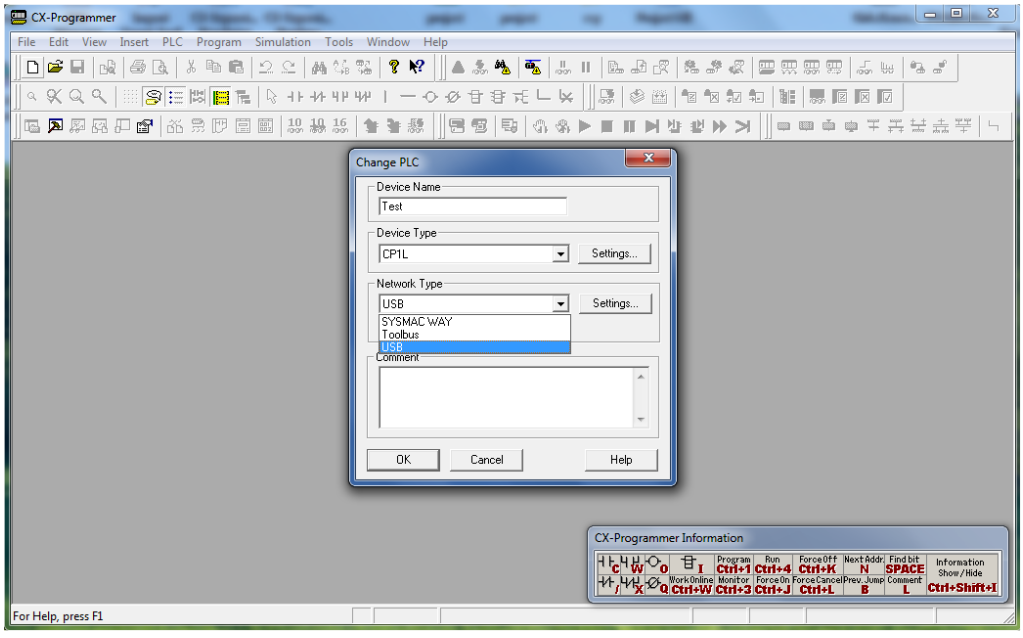


Figura 37– Escolha da Ligação do PLC na Janela "Change PLC" do CX-Programmer.

Agora, pode-se iniciar o programa, clicando-se em OK.

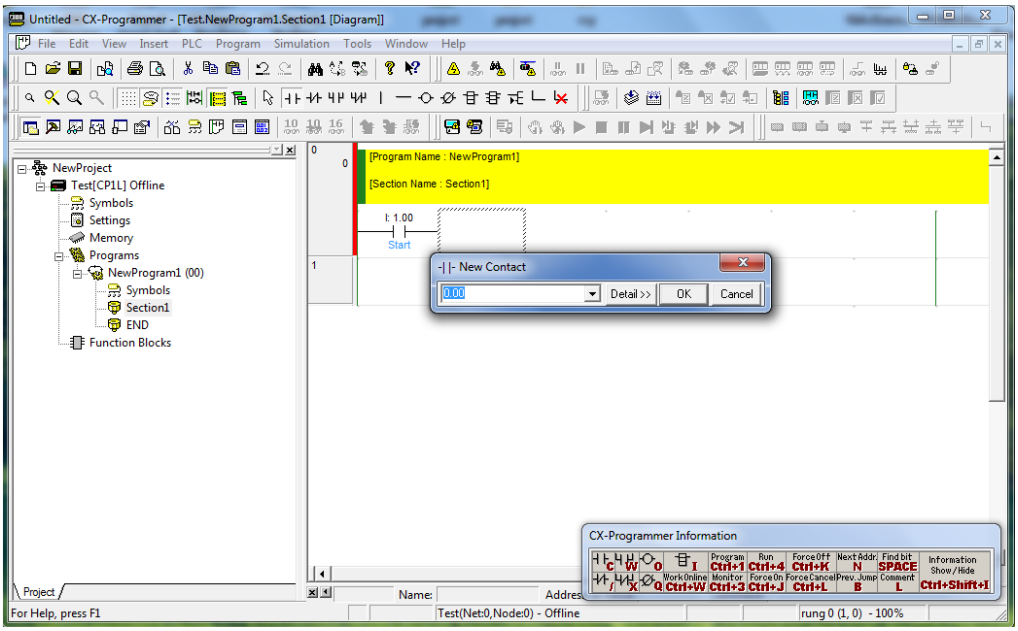


Figura 38 - Janela Principal do CX-Programmer.

Para o CP1L, tem-se as entradas/saídas desde 0.00 …. 0.99 até 1000.00 … 1000.99.

ANEXO B - UTILIZAÇÃO DO CX-DESIGNER PARA A ELABORAÇÃO DE SIMULAÇÕES VIRTUAIS

Utilização do CX-Designer para a elaboração de simulações virtuais.

O CX-Designer tem a função pode ser comparado a uma interface homem-máquina, e consegue comunicar com os PLCs virtuais.

Em função do funcionamento das simulações elaboradas no CX-Programmer e simuladas no CX-Simulator, diversos pontos-chave mudarão de estado.

Assim, o *software* CX-Designer irá descrever o desenrolar do programa em função dessas alterações de estado.

Por exemplo, se consideremos apenas um sensor de contato, com o ponto 0.00, este estará associado a uma imagem no CX-Designer, que para ser conectada devera-se seguir os seguintes passos:

Abrir-se o CX-Designer através do menu START ou no ambiente de trabalho.

Figura 39 - Atalho do CX-Designer.

Em seguida, o *software* abre-se, aparecendo uma pequena janela:

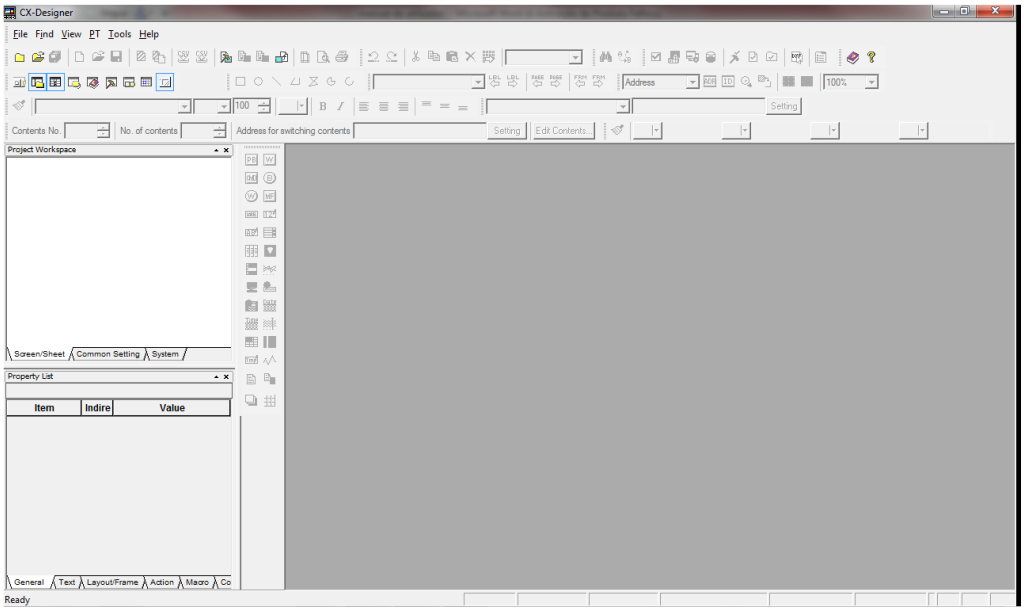


Figura 40 - Janela de trabalho do CX-Designer.

Em primeiro lugar, escolhe-se o “model” do CX-Designer, neste caso escolhe-se o NS15-TX0[]-V2.

EM seguida abre-se a menu bitmap.

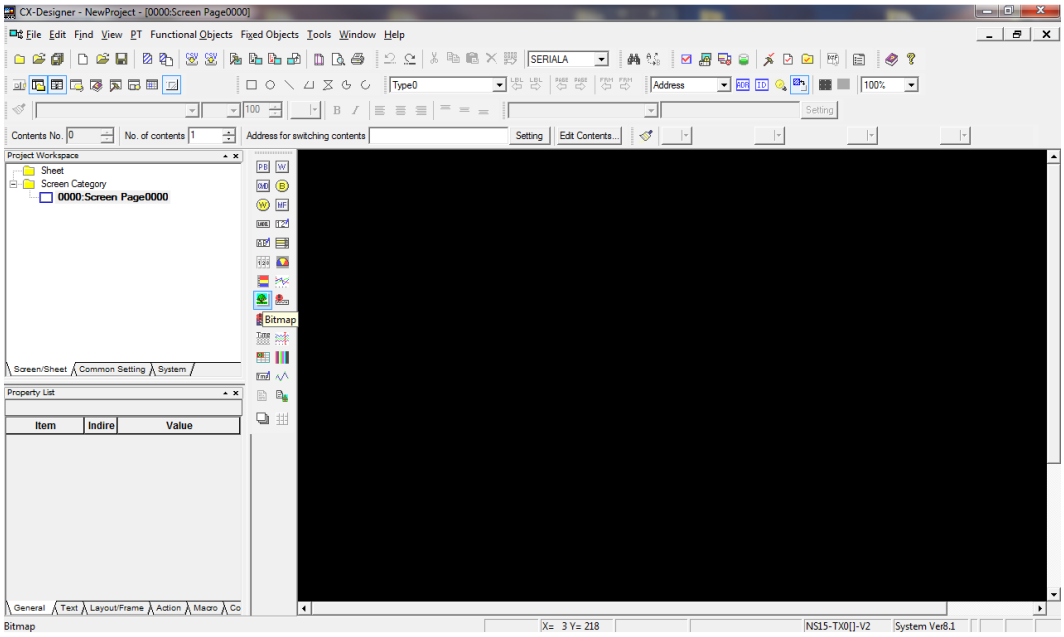


Figura 41 - Escolha do menu "bitmap" no CX-Designer.

Introduz-se uma imagem, através da tecla “browse”.

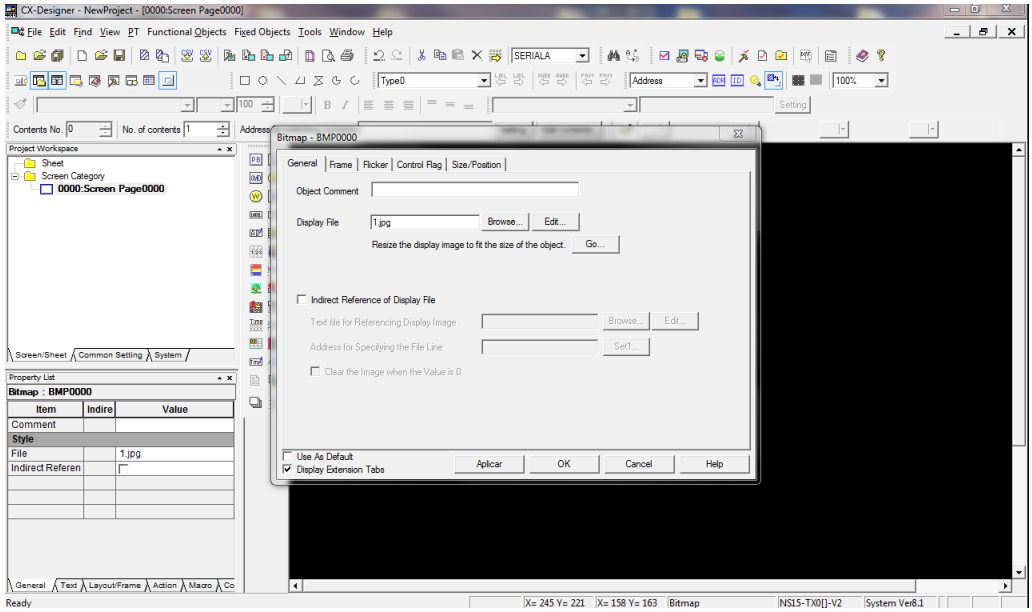


Figura 42 - Menu "bitmap" no CX-Designer

Para finalizar associa-se essa imagem ao endereço desejado “0.00”. clicando em “ControlFlag”.

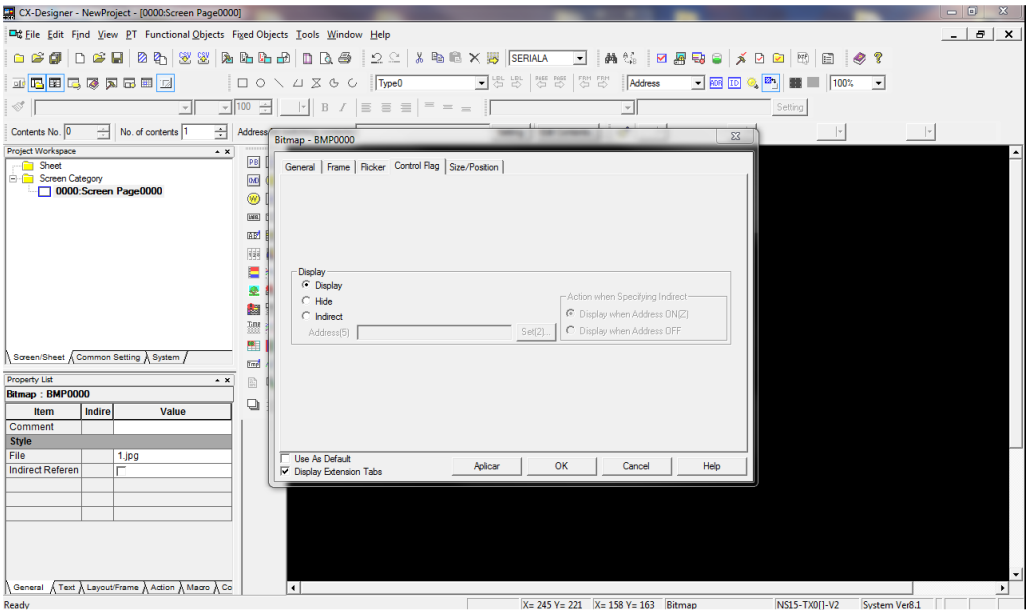


Figura 43 - Pasta "ControlFlag" no menu "bitmap".

Na função “display” escolhe-se a opção “indirect” e clica-se em “Set(2)…”

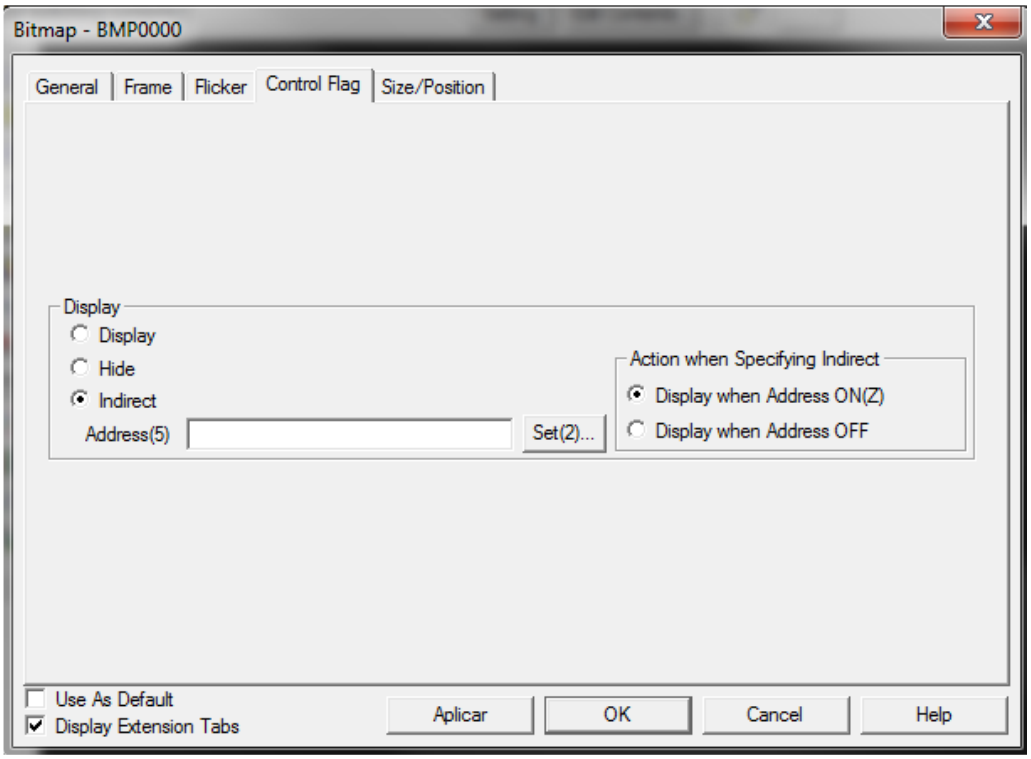


Figura 44 - Função "display" no menu "bitmap".

Altera-se a Area para “WorkArea”.

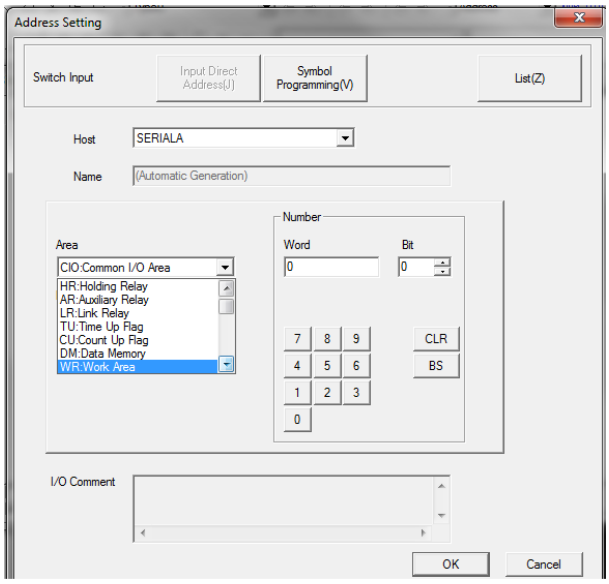


Figura 45 - "Adress Setting" (escolha de endereço) no menu "bitmap".

E na secção “Number” coloca-se o endereço desejado.

Se desejarmos transmitir uma informação ao CX-Programmar através de uma unidade de manipulação, é necessário seguir os seguintes passos:

Abre-se o CX-Designer como foi explicado anteriormente.

ANEXO C – ENUNCIADOS E SOLUÇÕES DOS EXERCÍCIOS

Enunciados e Solução dos Exercícios

C.1. Exercício 1 – Garra

Elabora o respetivo Grafcet de comando do sistema seguidamente apresentado. O funcionamento pretendido para o sistema é o seguinte:

- O carrinho que está sobre os carris tem uma pinça acoplada e tem como finalidade "pegar" em peças que chegam através do tapete B e colocá-las numa plataforma – acoplado a um cilindro vertical – que, ao descer, as coloca na direção do cilindro horizontal que as "empurra" para um tapete de saída C.
- A pinça descreve o movimento em "U" e só deve descer para pegar uma peça quando houver uma peça em posição para o efeito.
- Para simplificação, considere que a peça fica apertada após dois segundos decorridos desde o início da ordem para apertar.
- Considere que, quando o carro estiver do lado direito, com a pinça pronta para descer, e uma peça não aparecer, o carrinho deve ficar parado nessa posição.
- Considere um funcionamento do sistema que minimize o tempo de ciclo.
- A posição inicial é a apresentada na figura 46. Não é permitido que considere outra posição inicial.

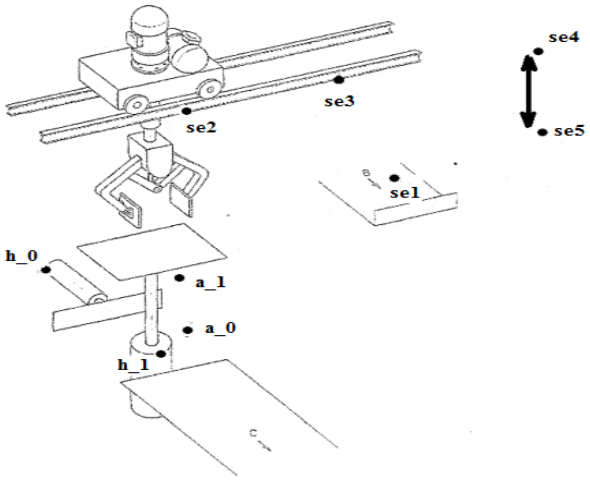


Figura 46 - Sistema e sensores constituintes do sistema a modelar.

Para simplificação, considere:

- D - Atuador que faz o carrinho mover-se para a direita.
- E - Atuador que faz o carrinho mover-se para a esquerda.
- DP - Atuador que faz descer a pinça.
- SP - Atuador que faz subir a pinça.
- AP - Pré-atuador (monoestável) que faz apertar a pinça.
- H - Pré-atuador (monoestável) que faz avançar o cilindro horizontal.
- V1 e V2 - Pré-atuador (biestável) responsável pelo avanço e recuo, respetivamente, do cilindro vertical.
- se1 - sensor que deteta a presença de peça no tapete B.
- se2 e se3 - sensores que detetam, respetivamente, o carrinho nos extremos à esquerda e à direita.
- se4 e se5 - sensores que detetam, respetivamente, o carrinho nos extremos superior e inferior.
- a_1 e a_0 - sensores de posição do cilindro vertical avançado e recuado respetivamente.
- h_1 e h_0 - sensores de posição do cilindro horizontal avançado e recuado respetivamente.

C.1.1. Modelo do controlador

A - Variáveis de Entrada e Saída

Tabela 8 - Descrição, Words e Bits de todas as Entradas do Exercício 1

Entrada	Descrição	Word & Bit
START	Botão de iniciação do sistema	0.00
STOP	Botão de paragem do sistema	0.01
se1	Sensor que deteta a presença de peça no tapete B	0.02
se2	Sensor que deteta o carrinho no extremo à esquerda	0.03
se3	Sensor que detetam carrinho no extremo à direita	0.04

se4	Sensor que detetam a pinça no extremo superior	0.05
se5	Sensor que detetam a pinça no extremo inferior	0.06
h_0	Sensor de posição do cilindro horizontal recuado	0.07
h_1	Sensor de posição do cilindro horizontal avançado	0.08
a_0	Sensor de posição do cilindro vertical recuado	0.09
a_1	Sensor de posição do cilindro vertical avançado	0.10

Tabela 9 - Descrição, Words e Bits de todas as Saídas do Exercício 1.

Saída	**Descrição**	**Word & Bit**
D	Atuador que faz o carrinho mover-se para a direita	100.00
E	Atuador que faz o carrinho mover-se para a esquerda	100.01
DP	Atuador que faz descer a pinça	100.02
SP	Atuador que faz subir a pinça	100.03
AP	Pré-atuador (monoestável) que faz apertar a pinça	100.04
H	Pré-atuador (monoestável) que faz avançar o cilindro horizontal	100.05
V2	Pré-atuador (biestável) responsável pelo recuo do cilindro vertical	100.06
V1	Pré-atuador (biestável) responsável pelo avanço do cilindro vertical	100.07

B - Especificação de comando

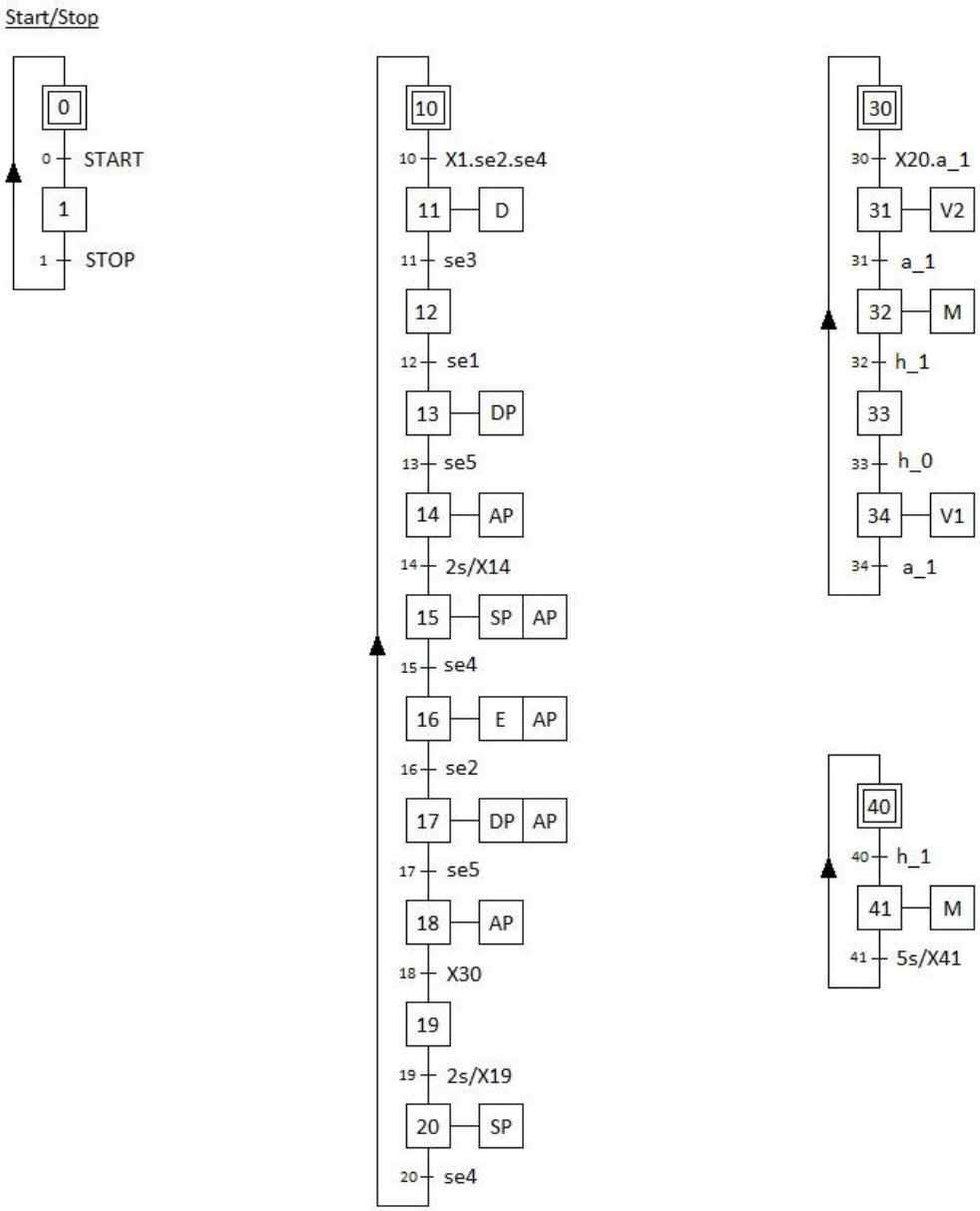


Figura 47 - Representação da modelização da parte de comando do exercício 1.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_0= /X0./X1

CT0 = X0.START

CT1 = X1.STOP

CTi_10 =/X10./X11./X12./X13./X14./X15./X16./X17./X18./X19./X20

CT10 = X10. X1.se2.se4

CT11 = X11.se3

CT12= X12.se1

CT13 = X13.se5

CT14 = X14.2s/X14

CT15 = X15.se4

CT16 = X16.se2

CT17 = X17.se5

CT18 = X18.X30

CT19 =X19.2s/X19

CT20 = X20.se4

CTi_30 =/X30./X31./X32./X33./X34

CT30 = X30.a_1

CT31 = X31.a_0

CT32 = X32.h_1

CT33 = X33.h_0

CT34 = X34.a_1

CTi_40 = /X40./X41

CT40 = X40. h_1

CT41 = X41.5s/X41

Em seguida elabora-se a atividade das etapas:

X0= CTi_0 + CT1 + X0./CT0

X1 = CT0 + X1./CT1

X10 = CTi_10 + CT20 + X10./CT10

X11 = CT10 + X11./CT11

X12 = CT11 + X12./CT12

X13 = CT12 + X13./CT13

X14 = CT13 + X14./CT14

X15 = CT14 + X15./CT15

X16 =CT15 + X16./CT16

X17 = CT16 + X17./CT17

X18 = CT17 + X18./CT18

X19 = CT18 + X19./CT19

X20 = CT19 + X20./CT20

X30 = CTi_30 + CT34 + X30./CT30

X31 =CT30 + X31./CT31

X32 = CT31 + X32./CT32

X33 = CT32 + X33./CT33

X34 = CT33 + X34./CT34

X40 = CTi_40 + X40./CT40

X41 = CT40 + X41./CT41

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

V2 = X31

H= X32

V1 = X34

D = X11

DP = X13 + X17

AP = X14 + X15 + X16 + X17 + X18

SP = X15 + X20

E = X16

M = X41

C.2. Exercício 2 – Encaixotamento de Maçãs

Pretende-se controlar a linha de encaixotamento de maçãs representada na figura 48:

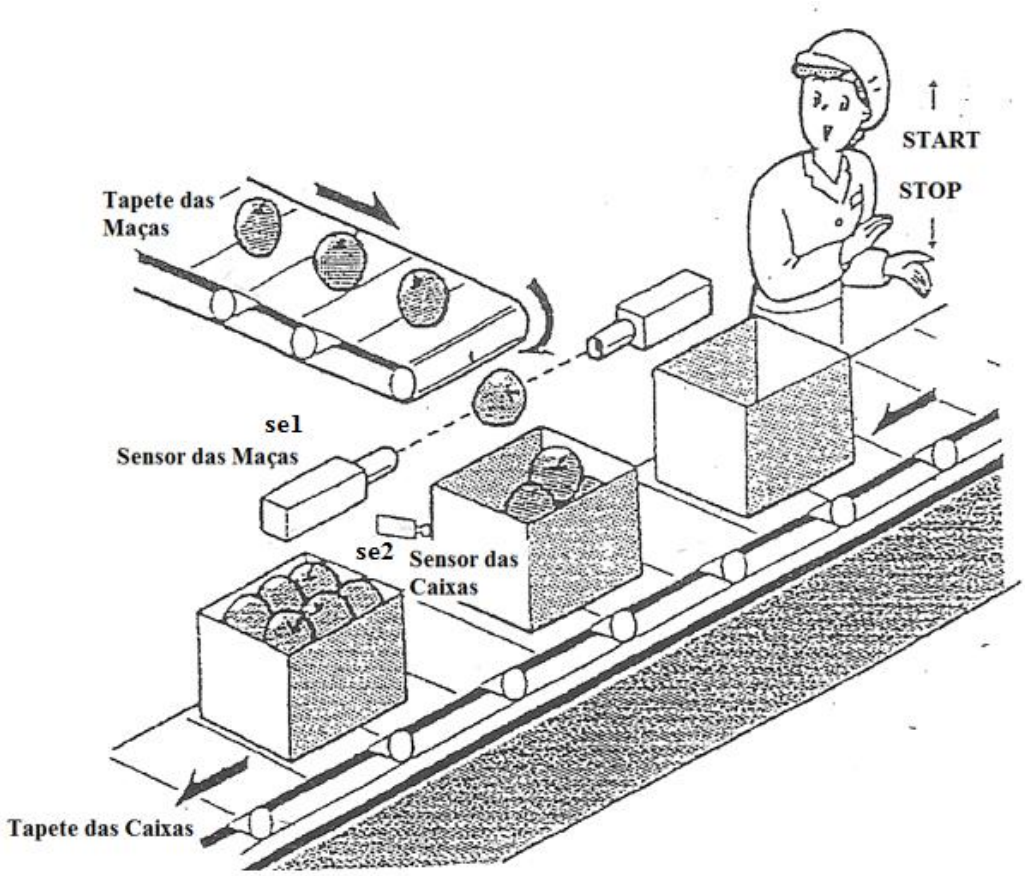


Figura 48 - Sistema de encaixotamento das maças e sensores constituintes do sistema a modelar.

Os tapetes possuem motores que os permitem rodar nas duas direções, dependendo da ordem emitida pelo controlador:

- M1 – Tapete das maçãs roda para a frente.
- M1_REC – Tapete das maçãs roda para trás.
- M2 – Tapete das caixas roda para a frente.
- M2_REC – Tapete das caixas roda para trás.

Ao sinal de START, o tapete das caixas entra em funcionamento. O sensor das caixas (se2) ao detetar uma caixa para este tapete e põe o das maçãs em funcionamento. O sensor das maçãs (se1) deteta as maçãs que entram na caixa. Após a caixa receber 10 maçãs, o tapete das maçãs para e o das caixas entra em funcionamento. O sinal de STOP pára todo o processo.

C.2.1. Modelo do controlador

A - Variáveis de Entrada e Saída

Tabela 10 - Descrição, Words e Bits de todas as Entradas do Exercício 2.

Entrada	Descrição	Word & Bit
START	Botão de inicialização do sistema	0.00
STOP	Botão de paragem do sistema	0.01
se1	Sensor que deteta as maçãs	0.02
se2	Sensor que deteta as caixas	0.03

Tabela 11 - Descrição, Words e Bits de todas as Saída do Exercício 2.

Saída	Descrição	Word & Bit
M1	Tapete das maçãs roda para frente	100.00
M1_REC	Tapete das maçãs roda para trás	100.01
M2	Tapete das caixas roda para a frente	100.02
M2_REC	Tapete das caixas roda para trás	100.03

B - Especificação de Comando

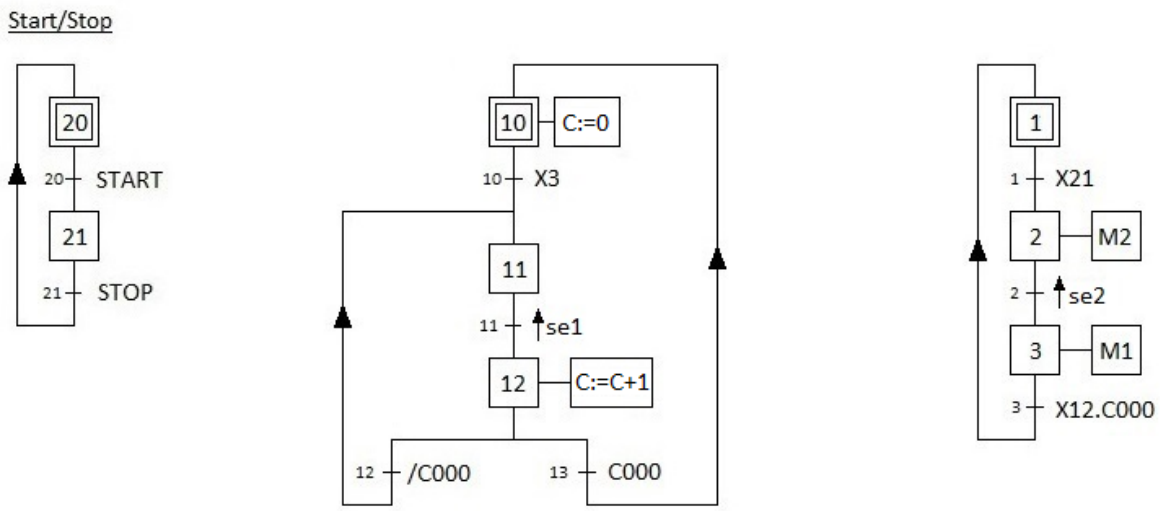


Figura 49 - Representação da modelização da parte de comando do exercício 2.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi _1= /X1./X2./X3

CT1 = X1.X21

CT2 = X2.↑se2

CT3 = X3.X12.C000

CTI_10 =/X10./X11./X12

CT10 = X10.X3

CT11 = X11. ↑se1

CT12= X12./C000

CT13 = X12.C000

CTi_20 =/X20./X21

CT20 = X20.START

CT21 = X21.STOP

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1 + CT3 + X1./CT1

X2 = CT1 + X2./CT2

X3 = CT2 + X3./CT3

X10 = CTi_10 + CT13 + X10./CT10

X11 = CT10 + CT12 + X11./CT11

X12 = CT11 + X12./(CT12 + CT13)

X20 = CTi_20 + CT21 + X20./CT20

X21 = CT20 + X21./CT21

X30 = CTi_30 + CT41 + X30./CT30

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

M2 = X2

M1 = X3

C.3. Exercício 3 – Barreira automática

O sistema automatizado representado na figura 50 controla a entrada de veículos para um parque de estacionamento. Ele é constituído por uma barreira acionada por um motor com dois sentidos de rotação: movimento de subida (M1S) e movimento de descida (M1D). Além disso, existe também um conjunto de quatro sensores:

- Dois para detetar a presença de viatura quando se encontra na entrada da barreira (se1) e quando a atravessa (se2);
- Dois para a deteção da posição da barreira, sendo um para detetar quando se encontra totalmente subida (sbs) e outro para quando estiver totalmente descida (sbd).

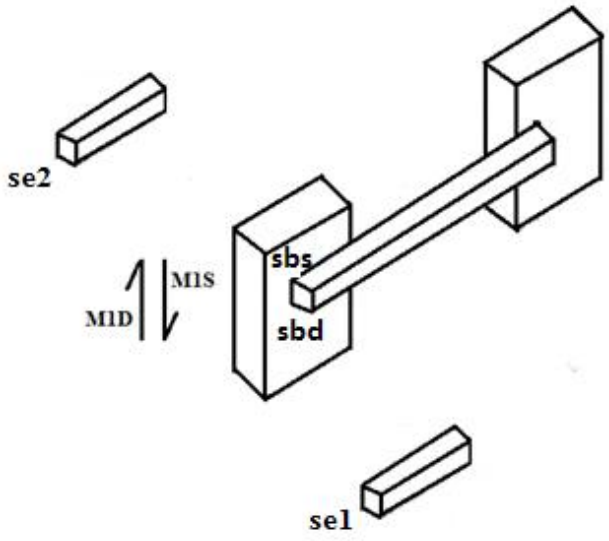


Figura 50 - Barreira automática e sensores constituintes do sistema a modelar.

Quando se aproxima uma viatura (se1 atuado), a barreira deve subir e deve permanecer levantada até que não seja detetada viatura por se1. Depois disto, a barreira deve descer, mas se porventura, nalgum instante durante a descida, o sensor se1 detetar a presença de uma nova viatura, a barreira deve subir imediatamente para não a danificar.

Considere que o sistema possui um relógio associado a um sistema elétrico, que emite um sinal elétrico (EL1) sempre que são 06:00h e outro sinal elétrico (EL2) sempre que são 22:00h. Utilize estes dois sinais para fazer automaticamente a gestão de tempos da "operacionalidade" da barreira.

Com a finalidade de tratamento estatístico, no final de cada dia deve ser possível saber quantos carros entraram no parque. Para este efeito devem ser contabilizadas as vezes que o sensor se2 liga e desliga.

C.3.1. Modelo do controlador

A - Variáveis de Entrada e Saída

Tabela 12 - Descrição, Words e Bits de todas as Entradas do Exercício 3.

Entrada	Descrição	Word & Bit
se1	Sensor que deteta os carros antes da barreira	0.02
sbd	Sensor que deteta a barreira descida	0.03
sbs	Sensor que deteta a barreira subida	0.04
se2	Sensor que deteta os carros depois da barreira	0.05

Tabela 13 - Descrição, Words e Bits de todas as Saídas do Exercício 3.

Saída	Descrição	Word & Bit
M1S	Movimento de subida da barreira	100.00
M1D	Movimento de descida da barreira	100.01
EL1	Sinal elétrico do início do período de operação	100.02
EL2	Sinal elétrico do fim do período de operação	100.03

B - Especificação de Comando

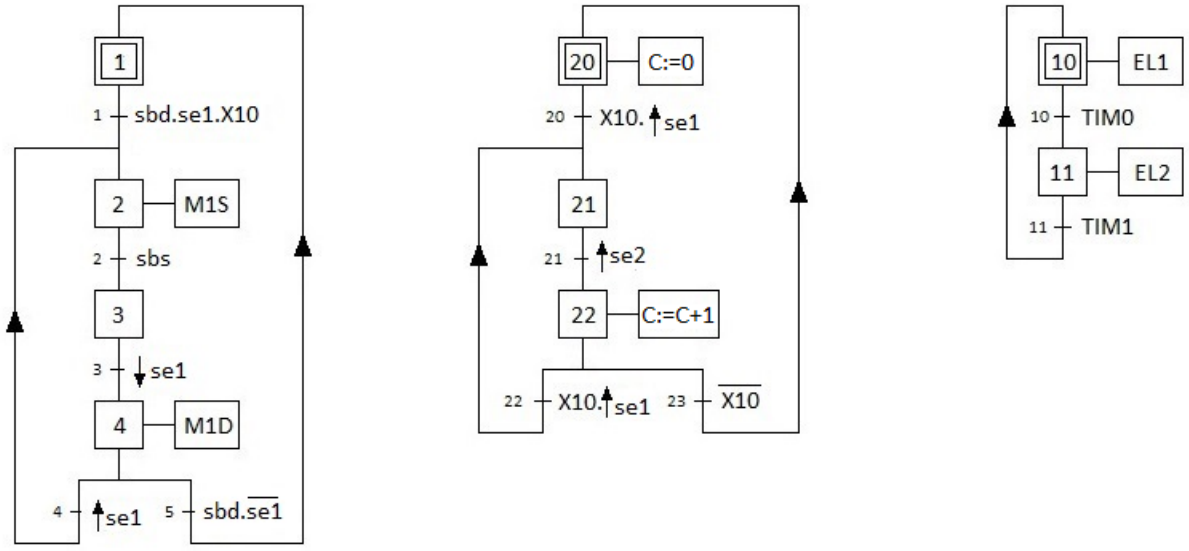


Figura 51 - Representação da modelização da parte de comando do exercício 3.

NOTA: por motivos académicos os Timmers 0 e 1 tem um tempo variável e por isso não têm um tempo especificado no Grafcet de comando.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi _1= /X1./X2./X3./X4

CT1 = X1.sbd.se1.X10

CT2 = X2.sbs

CT3 = X3.↓se1

CT4 = X4.↑se1

CT5 = X4.sbd./se1

CTi_10 =/X10./X11

CT10 = X10.TIM0

CT11 = X11.TIM1

CTi_20 =/X20./X21./X22

CT20 = X20.X10.↑se1

CT21 = X21. ↑se2

CT22 = X22.X10.↑se1

CT23 = X22./X10

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1+ CT5 + X1./CT1

X2 = CT1 + CT4+ X2./CT2

X3 = CT2 + X3./CT3

X4 = CT3 + X4./(CT4 + CT5)

X10 = CTi_10 + CT11 + X10./CT10

X11 = CT10 + X11./CT11

X20 = CTi_20 + CT23 + X20./CT20

X21 = CT20 + CT22 + X21./CT21

X22 = CT21 + X22./(CT22 + CT23)

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

M1S = X2

M1D = X4

EL1 = X10

EL2 = X11

C.4. Exercício 4 - Portão da Garagem

Pretende-se automatizar um portão de garagem. No interior e no exterior da garagem existem botoneiras de comando para permitir a abertura e fecho do portão. A atuação da botoneira de subida origina o movimento do portão até à sua abertura total, correspondente ao sensor "b" acuado. A atuação sobre a botoneira de descida permite a movimentação do portão até a seu fecho completo, correspondendo ao sensor "a" atuado. Durante o movimento de abertura u fecho do portão, um impulso sobre a botoneira de stop permite a paragem do portão. Para retomar o movimento de subida ou descida, é bastante atuar a botoneira respetiva.

O funcionamento descrito deve conseguir indistintamente pelo lado interior ou exterior da garagem e, durante a movimentação do portão, a lâmpada "L" deve permanecer acesa, apagando-se sempre que o portão se encontra parado.

As botoneiras e sensores são todos normalmente abertos e o motor de acionamento do portão é trifásico e comandado por contactores.

Considere a parte interior prioritária.

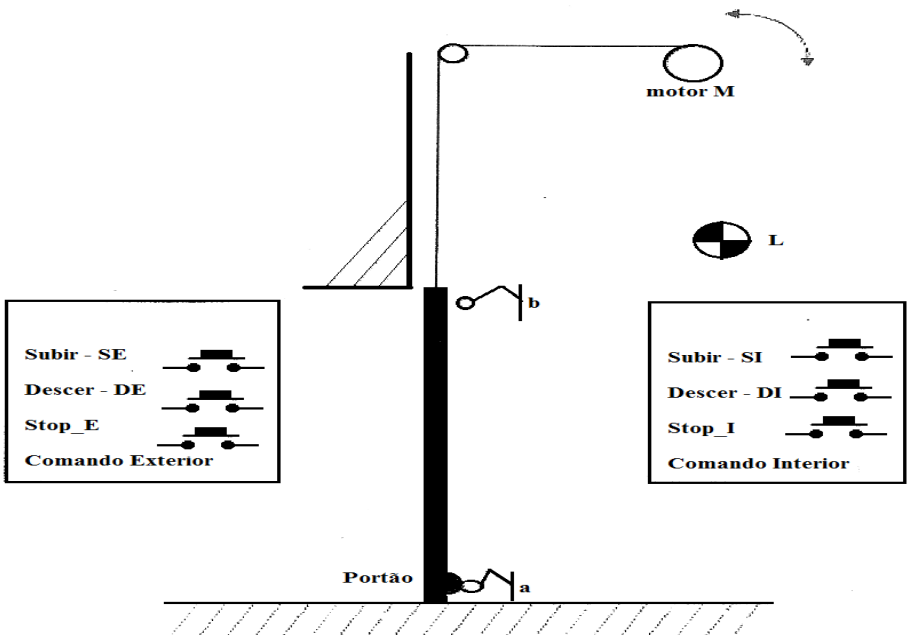


Figura 52 - Portão da garagem e sensores constituintes do sistema a modelar.

C.4.1. Modelo do controlador

A - Variáveis de Recetividades e Ações

Tabela 14 - Descrição, Words e Bits de todas as Entradas do Exercício 4.

Entrada	Descrição	Word & Bit
DE	Botão de descida exterior	0.00
SE	Botão de subida exterior	0.01
STOP_E	Botão de paragem exterior	0.02
b	Sensor que deteta o portão subido	0.03
a	Sensor que deteta o portão descido	0.04
DI	Botão de descida interior	0.05
SI	Botão de subida interior	0.06
STOP_I	Botão de paragem interior	0.07

Tabela 15 - Descrição, Words e Bits de todas as Saídas do Exercício 4.

Saída	Descrição	Word & Bit
MS	Movimento de subida do portão	100.00
MD	Movimento de descida do portão	100.01
L	Ligar Lâmpada	100.02

B - Especificação de Comando

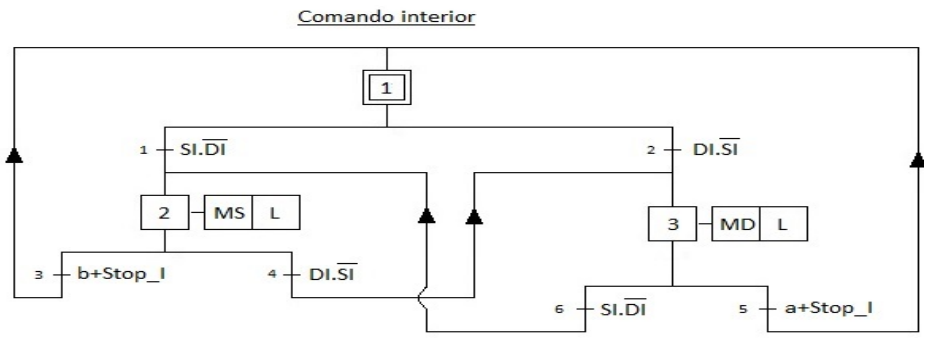


Figura 53 - Representação da modelização da parte de comando interno do exercício 4.

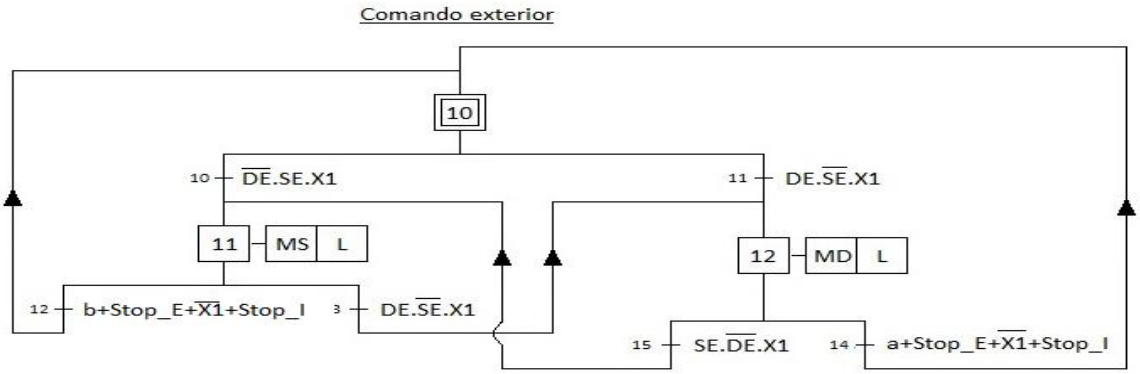


Figura 54 - Representação da modelização da parte de comando exterior do exercício 4.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_1= /X1./X2./X3

CT1 = X1.SI./DI

CT2 = X1.DI./SI

CT3 = X2.(b + STOP_I)

CT4 = X2.DI./SI

CT5 = X3.(a + STOP_I)

CT6 = X3.SI./DI

CTi_10 = /X10./X11./X12

CT10 = X10./DE.SE.X1

CT11 = X10./SE.DE.X1

CT12= X11.(b + STOP_E + /X1 + STOP_I)

CT13 = X11.DE./SE.X1

CT14 = X12.(a + STOP_E + /X1 + STOP_I)

CT15 = X12.(SE./DE.X1)

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1 + CT3 + CT5 + X1./(CT1 + CT2)

X2 = CT1+ CT6 + X2./(CT3 + CT4)

X3 = CT2 + CT4 + X3./(CT5 + CT6)

X10 = CTi_10 + CT12 + CT14 + X10./(CT10 + CT11)

X11 = CT10 + CT15 + X11./(CT12 + CT13)

X12 = CT11 + CT13 + X12./(CT14 + CT15)

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

MS = X2 + X11

MD = X3 + X12

L = X2 +X3 + X11 + X12

C.5. Exercício 5 - Passagem de Nível

Pretende-se desenvolver o programa de comando de uma barreira automatizada para uma passagem de nível. Esta move-se para baixo (MD) sempre que o comboio se dirige à passagem de nível e para cima (MU) sempre que este se afasta da passagem de nível. O comboio pode vir de ambas as direções.

A linha férrea possui três sensores (tabela 16) que detetam o comboio: sA, sB e sC como demonstrado na figura 55. Também existem dois sensores para a deteção da barreira descida (sbd) e da barreira subida (sbu).

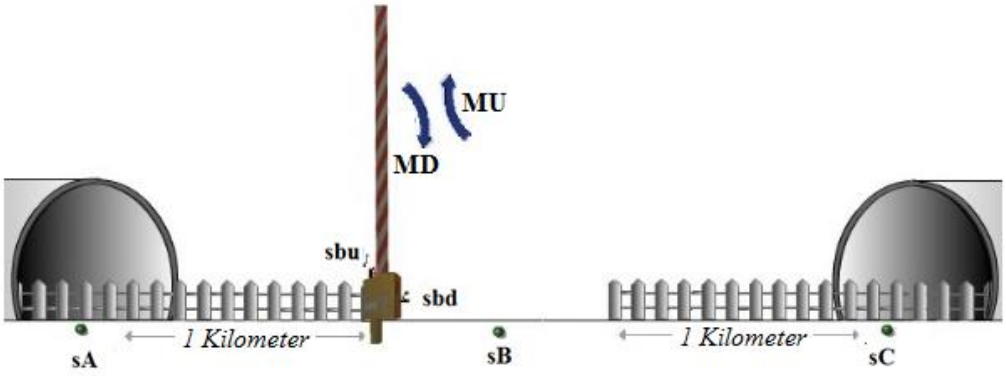


Figura 55 - Esquema da passagem de nível.

C.5.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 16 - Descrição, Words e Bits de todas as Entradas do Exercício 5.

Entrada	Descrição	Word & Bit
sA	Comboio em cima do sensor A	0.00
sB	Comboio em cima do sensor B	0.01
sC	Comboio em cima do sensor C	0.02
sbd	Barreira Descida	0.03
sbu	Barreira Subida	0.04

Tabela 17 - Descrição, Words& Bits de todas as Saídas do Exercício 5.

Saída	Descrição	Word & Bit
MU	Barreira move-se para cima	100.00
MD	Barreira move-se para baixo	100.01

B - Especificação de Comando

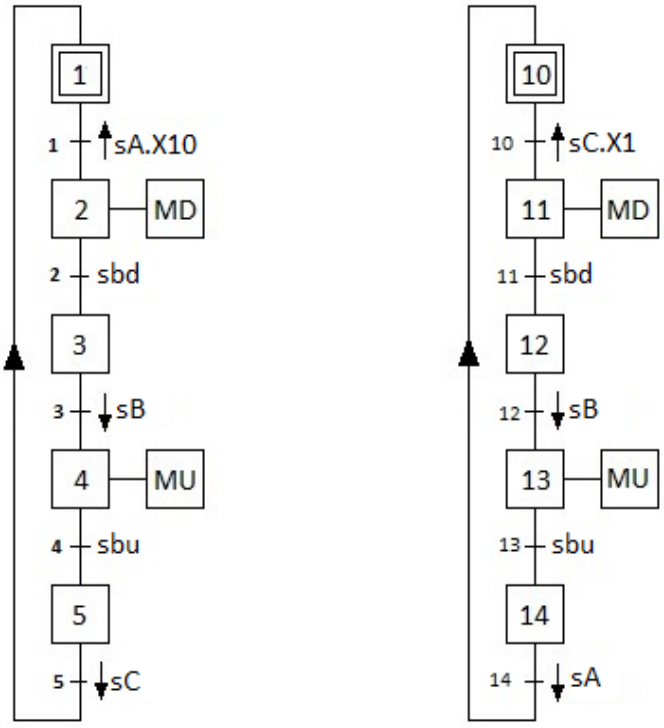


Figura 56 - Representação da modelização da parte de comando do exercício 5.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_1 = /X1./X2./X3./X4./X5

CTi_10 = /X10./X11./X12./X13./X14

CT1 = X1.↑sA. X10

CT2 = X2.sbd

CT3 = X3.↓sB

CT4 = X4.sbu

CT5 = X5.↓sC

CT10 = X10.↑sC. X1

CT11 = X11.sbd

CT12= X12.↓sB

CT13 = X13.sbu

CT14 = X14.↓sA

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1+ CT5 + X1./CT1

X2 = CT1 + X2./CT2

X3 = CT2 + X3./CT3

X4 = CT3 + X4./CT4

X5 = CT4 + X5./CT5

X10 = CTi_10 + CT13 + X10./CT10

X11 = CT10 + X11./CT11

X12 = CT11 + X12./CT12

X13 = CT12 + X13./CT13

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

MD = X2 + X11

MU = X4 + X13

C.6. Exercício 6 - Aspirador

Tendo em conta o sistema da figura 57, elabore a especificação do respetivo comando.

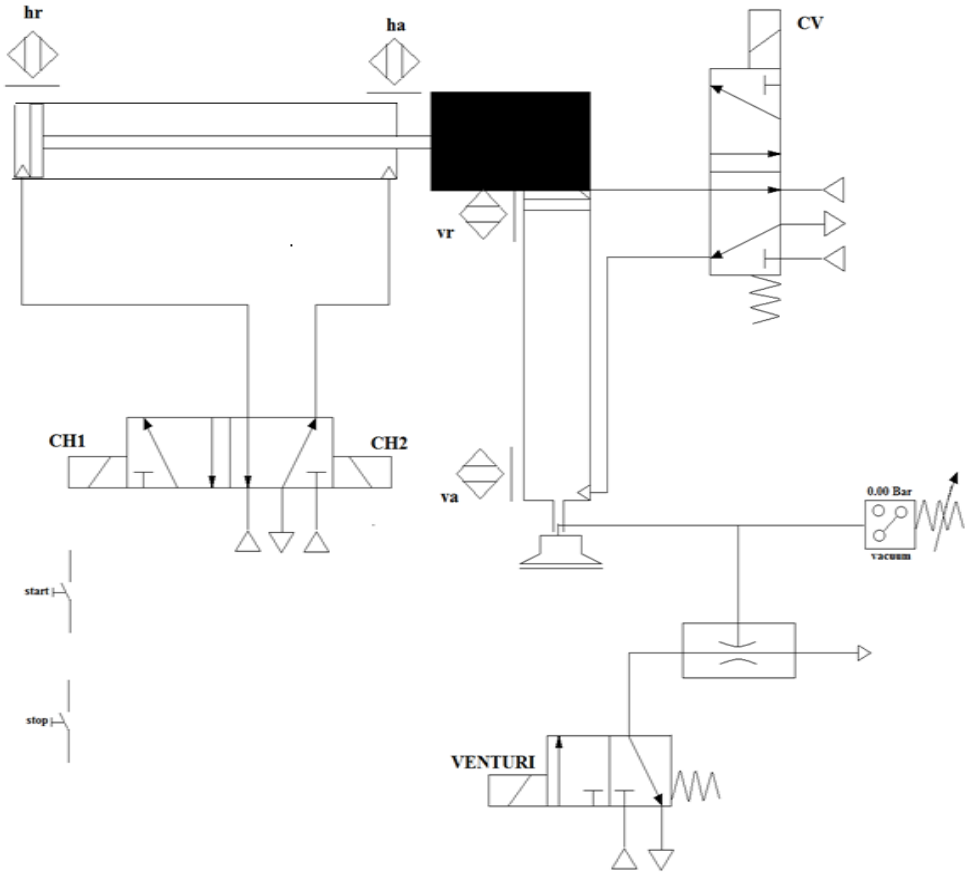


Figura 57 - Sistema automatizado de aspiração

Em Grafcet, tendo em conta o segundo percurso:

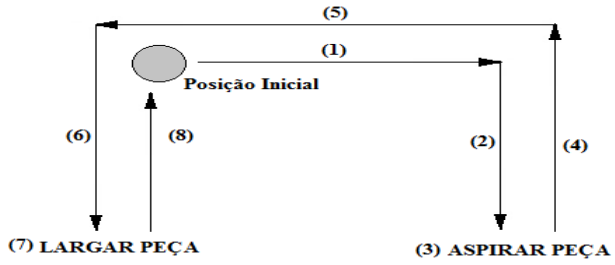


Figura 58 - Percurso percorrido pelo sistema automatizado.

O sistema possui um sensor (tabela 18) que deteta a presença de peça pronta a ser posta no sistema: se1. Também existem quatro sensores de final de curso dos cilindros pneumáticos existentes no sistema: hr, ha, vr, va que representas o cilindro horizontal recuado, avançado e o cilindro vertical recuado e avançado respetivamente. A válvula que comanda o cilindro

horizontal é biestável enquanto as que comandam o cilindro vertical e o aspirador são monoestáveis.

As ordens CH1 e CH2 fazem o cilindro avançar e recuar respetivamente; CV ordena o avanço do cilindro vertical; V ordena a aspiração da peça. É de salientar também que para a peça ser aspirada são necessários dois segundos assim como para que a ventosa largue esta.

NOTA: Deve ser considerado um Grafcet para cada cilindro e outro para a aspiração.

C.6.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 18 - Descrição, Words e Bits de todas as Entradas do Exercício 6.

Entrada	Descrição	Word & Bit
START	Botão de iniciação do sistema	0.00
hr	Cilindro horizontal recuado	0.01
ha	Cilindro horizontal avançado	0.02
vr	Cilindro vertical recuado	0.03
va	Cilindro vertical avançado	0.04
STOP	Botão de paragem do sistema	0.05
se1	Sensor de deteção de peças prontas para ser aspiradas pelo sistema	0.06

Tabela 19 - Descrição, Words e Bits de todas as Saídas do Exercício 6.

Saída	Descrição	Word & Bit
CH1	Avanço do Cilindro Horizontal	100.00
CH2	Recuo do Cilindro Horizontal	100.01
CV	Avanço do Cilindro Vertical	100.02
V	Aspiração das Peças	100.03

B - Especificação de Comando

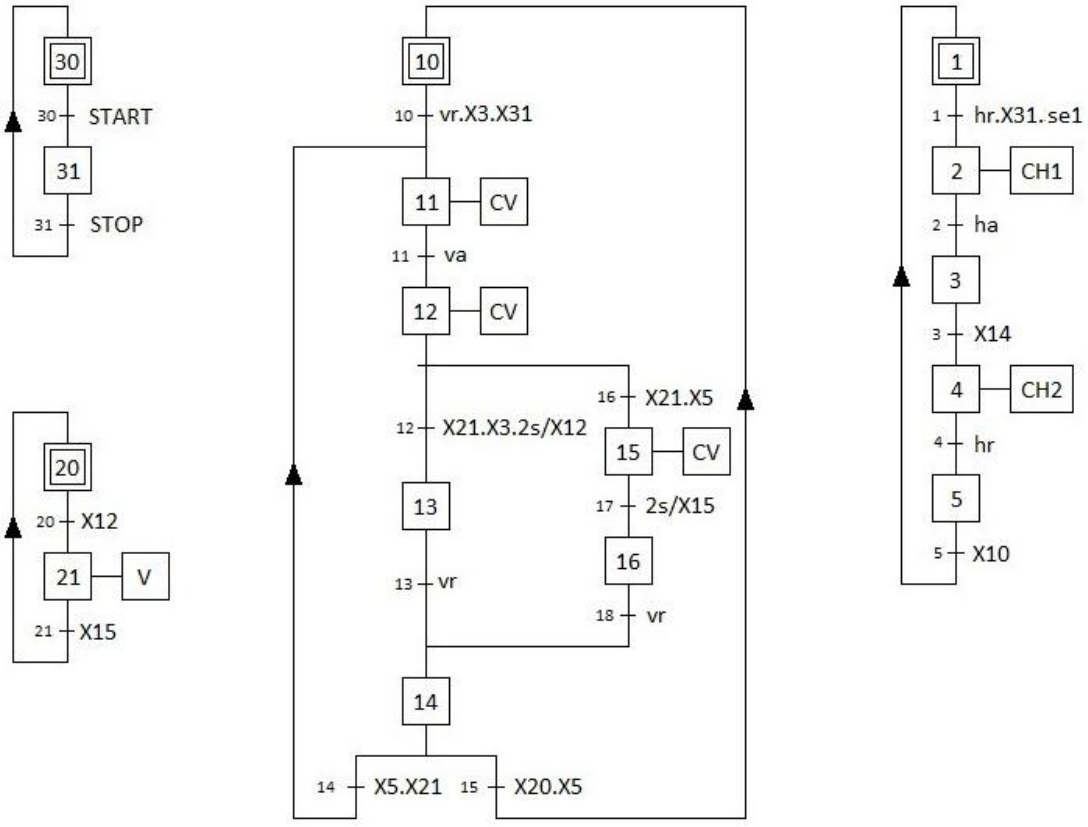


Figura 59 - Representação da modelização da parte de comando do exercício 6.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_1 = /X1./X2./X3./X4./X5

CT1 = X1.hr.X31.se1

CT2 = X2.ha

CT3 = X3.X14

CT4 = X4.hr

CT5 = X5.X10

CTi_10 = /X10./X11./X12./X13./X14./X15./X16

CT10 = X10.vr.X3.X31

CT11 = X11.va

CT12= X12.X21.X3.2s/X12

CT13 = X13.vr

CT14 = X14.X5.X21

CT15 = X14.X20.X5

CT16 = X12.X21.X5

CT17 = X15.2s/X15

CT18 = X16.vr

CT20 = X20.X12

CTi_20 = /X20./X21

CT21 = X21.X15

CTi_30 = /X30./X31

CT30 = X30.START

CT31 = X31.STOP

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1+ CT5 + X1./CT1

X2 = CT1 + X2./CT2

X3 = CT2 + X3./CT3

X4 = CT3 + X4./CT4

X5 = CT4 + X5./CT5

X10 = CTi_10 + CT15 + X10./CT10

X11 = CT10 + CT14 + X11./CT11

X12 = CT11 + X12./(CT12 + CT16)

X13 = CT12 + X13./CT13

X14 = CT13 + CT17 + X14./(CT14 + CT15)

X15 = CT16 + X15./CT17

X16 = CT17 + X16./CT18

X20 = CTi_20 + CT21 + X20./CT21

X21 = CT20 + X21./CT21

X30 = CTi_30 + CT31 + X30./CT30

X31 = CT30 + X31./CT31

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

CH1 = X2

CH2 = X4

CV = X11 + X12 + X15

V = X21

C.7. Exercício 7 – Misturadora de Solução Química

A figura 60 representa uma instalação para preparação de uma solução química, na qual intervêm o produto A, o produto B e o solvente S.

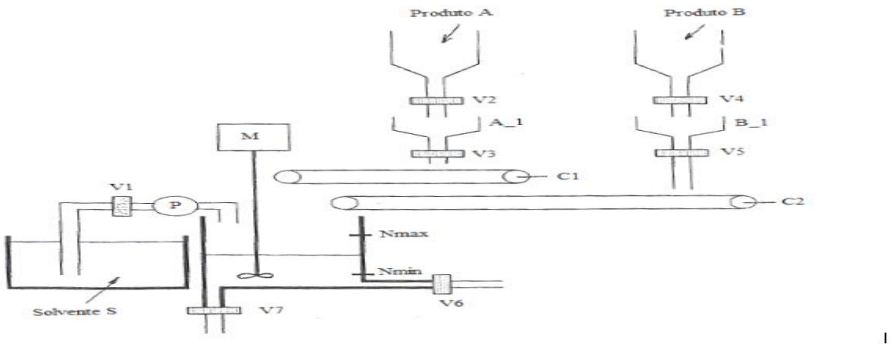


Figura 60 - Instalação de preparação da solução química e respetivos sensores.

O funcionamento da instalação é o seguinte:

- Se o nível da solução, na cuba misturadora, for inferior ao mínimo (Nmin), e o operador acionar o botão de arranque (START), as válvulas V1, V2 e V4 abrem-se e o motor (P), da bomba de aspiração do solvente, entra em funcionamento.
- Quando uma determinada quantidade dos produtos A e B tiver sido lançada sobre as básculas A_1 e B_1, respetivamente, as válvulas V2 e V4 fecham-se. Entretanto quando se atingir o nível máximo (Nmax) de solvente na cuba misturadora, o motor (P) da bomba é desligado, assim como é também fechada a válvula V1.
- Quando as três condições referidas no ponto anterior se verificarem, o motor M da hélice misturadora entra em funcionamento, os motores dos tapetes transportadores recebem as ordens C1_AV e C2_AV, iniciando também o seu funcionamento e, simultaneamente, abrem-se as válvulas V3 e V5.
- Ao fim de 10 segundos as válvulas V3 e V5 são fechadas, assim como ambos os tapetes rolantes param em simultâneo.
- O motor M, da hélice misturadora, mantém-se em funcionamento por mais 20 segundos, ao fim dos quais parará, abrindo-se a válvula V6 para enviar a solução para o posto de utilização, mantendo-se aberta ao longo de 15 segundos.
- Se durante esses 20 segundos o operador considerar que a solução obtida não é satisfatória, acionará o botão REG, que provocará a paragem do motor M e a abertura da válvula V7, escoando-se a solução para o contentor apropriado.

- Quando o nível mínimo (Nmin) de solução na cuba for atingido, a válvula V6 ou V7, conforme a evolução havida, fechará e retornar-se-á o estado inicial da máquina.

É admitido que os produtos A e B encontram-se sempre disponíveis nos seus postos de armazenamento e que todas as válvulas são monoestáveis de retorno por mola.

C.7.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 20 - Descrição, Words e Bits de todas as Entradas do Exercício 7.

Entrada	Descrição	Word & Bit
START	Botão de inicialização do sistema	0.00
REG	Botão de paragem de sistema	0.01
Nmin	Sensor que deteta o nível mínimo de solução	0.02
Nmax	Sensor que deteta o nível máximo de solução	0.03
A_1	Sensor que deteta o peso do produto A	0.04
B_1	Sensor que deteta o peso do produto B	0.05

Tabela 21 - Descrição, Words e Bits de todas as Saídas do Exercício 7.

Saída	Descrição	Word & Bit
V1	Abertura da Válvula 1	100.00
V2	Abertura da Válvula 2	100.01
V3	Abertura da Válvula 3	100.02
V4	Abertura da Válvula 4	100.03
V5	Abertura da Válvula 5	100.04
V6	Abertura da Válvula 6	100.05
V7	Abertura da Válvula 7	100.06
P	Atuação da bomba hidráulica	100.07
M	Atuação do motor da misturadora	101.01
C1_AV	Tapete de transporte do produto A roda para frente	101.03
C1_REC	Tapete de transporte do produto A roda para trás	101.04
C2_AV	Tapete de transporte do produto B roda para frente	101.05
C2_REC	Tapete de transporte do produto B roda para trás	101.06

B - Especificação de Comando

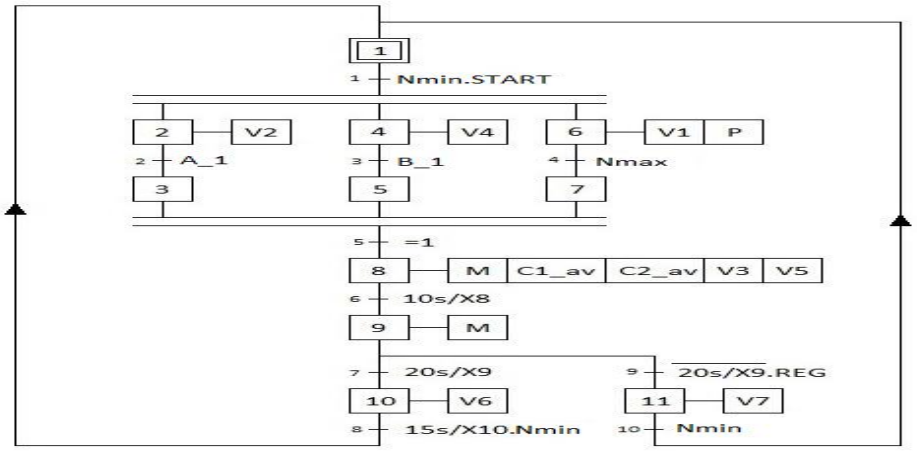


Figura 61 - Representação da modelização da parte de comando do exercício 7.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_1 =
/X1./X2./X3./X4./X5./X6./X7./X8./X9./X10./X11
CT1 = X1.Nmin.START
CT2 = X2.A_1
CT3 = X4.B_1
CT4 = X6.Nmax
CT5 = X3.X5.X7
CT6 = X8.10s/X8
CT7 =X9.20s/X9
CT8 = X10.15s/X10.Nmin
CT9 =X9./(20s/X9).REG
CT10 = X11.Nmin

Em seguida elabora-se a atividade das etapas:
X1 = CTi_1 + CT8 + CT10 + X1./CT1
X2 = CT1 + X2./CT2
X3 = CT2 + X3./CT5
X4 = CT1 + X4./CT3
X5 = CT3 + X5./CT5
X6 = CT1 + X6./CT4
X7 = CT4 + X7./CT5
X8 = CT5 + X8./CT6
X9 = CT6 + X9./(CT7 + CT9)
X10 = CT7 + X10./CT8
X11 = CT9 + X11./CT10

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

V1 = X6
V2 = X2
V3 = X8
V4 = X4
V5 = X8
V6 = X10
V7= X11
P = X6
M = X8 + X9
C1_AV = X8
C2_AV = X8

C.8. Exercício 8 - Barreira com Operador

Elabore o Grafcet de comando do sistema seguidamente apresentado:

Suponha que, na empresa onde irá exercer a sua profissão após a conclusão da licenciatura, é necessário controlar a entrada de viaturas. O sistema automatizado adotado é constituído por uma barreira acionada por um motor com dois sentidos de rotação: Movimento de subida (M1S) e movimento de descida (M1D). Além disso, existe também um conjunto de quatro sensores:

- Dois para detetar a presença de viatura quando se encontra na entrada da barreira (se1) e quando a atravessa (se2).
- Dois para a deteção da posição da barreira, sendo um para detetar quando se encontra totalmente subida (sbs) e outro para quando estiver totalmente descida (sbd).

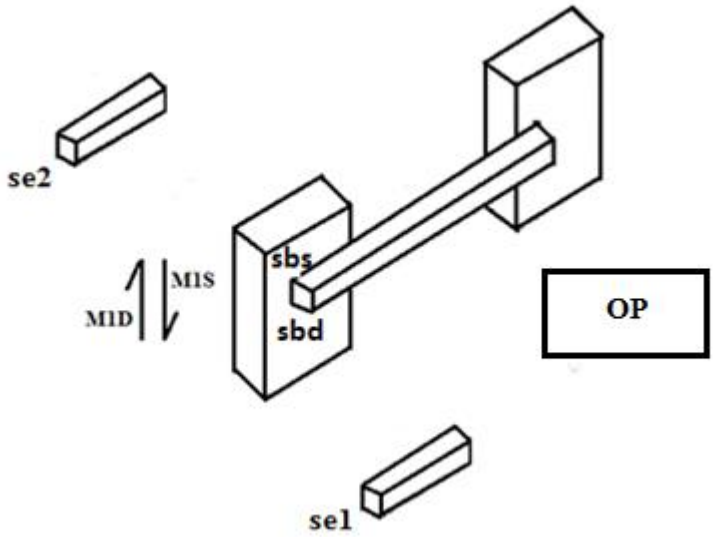


Figura 62 -Sistema de Barreira e respetivos sensores.

O funcionamento pretendido é o seguinte:

- O controlo da entrada de viaturas no recinto da empresa é feito por um operador que se encontra num local onde pode diretamente ver a barreira e uma possível viatura que se prepare para entrar. Para tal utiliza o botão OP.
- Quando se aproxima uma viatura deve acender-se uma luz amarela (LA) no local onde se encontra o operador.
- Depois de observar a luz amarela o operador decide se a viatura deve, ou não, entrar. Se decidir que ela deve entrar, deve pressionar o botão OP e a barreira deve subir. Se

decidir que a viatura não deve entrar, não pressiona o botão OP e a luz amarela deve apagar-se após 10 segundos.

- Depois de a barreira subir deve permanecer levantada até que não seja detetada viatura em se1.
- Depois de não ser detetada viatura em se1 a barreira deve descer, mas se porventura, nalgum instante durante a descida, o sensor se1 detetar a presença de uma nova viatura, a barreira deve subir imediatamente para não danificar a viatura, mas deve acender-se uma luz vermelha (LV) que permita ao operador aperceber-se que está a entrar alguma viatura sem que ele tenha dado ordem para tal. Após 10 segundos, essa luz deve apagar-se.
- Em qualquer das situações anteriores, logo que não seja detetada viatura em se1, a barreira deve descer até à posição horizontal (sbd atuado) e assim deve permanecer até aparecer outra viatura.

C.8.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 22 - Descrição, Words e Bits de todas as Entradas do Exercício 8.

Entrada	Descrição	Word & Bit
se1	Sensor que deteta os carros antes da barreira	0.02
sbd	Sensor que deteta a barreira descida	0.03
sbs	Sensor que deteta a barreira subida	0.04
se2	Sensor que deteta os carros depois da barreira	0.05
OP	Botão de acionamento da barreira	0.06

Tabela 23 - Descrição, Words e Bits de todas as Saídas do Exercicio 8.

Saída	Descrição	Word & Bit
M1S	Movimento de subida da barreira	100.00
M1D	Movimento de descida da barreira	100.01
LV	Luz vermelha	100.02
LA	Luz amarela	100.03

B - Especificação de controlo

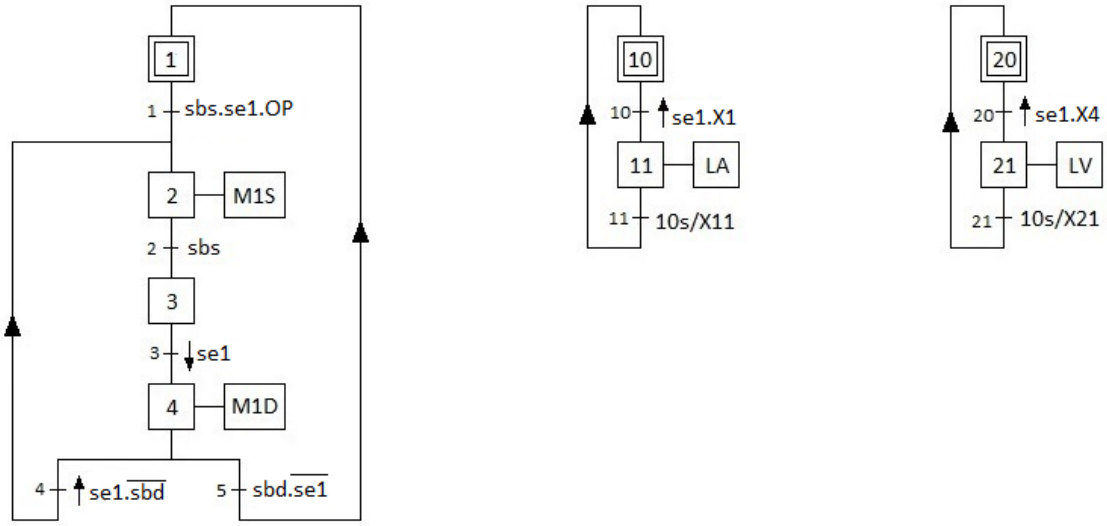


Figura 63 - Representação da modelização da parte de comando do exercício 8.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi _1= /X1./X2./X3./X4

CT1 = X1.sbd.se1.OP

CT2 = X2.sbs

CT3 = X3.↓se1

CT4 = X4.↑se1./sbd

CT5 = X4.sbd./se1

CTi_10 =/X10./X11

CT10 = X10.↑se1.X1

CT11 = X11.10s/X11

CTi_20 =/X20./X21

CT20 = X20.X10.↑se1

CT21 = X21.10s/X21

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1+ CT5 + X1./CT1

X2 = CT1 + CT4+ X2./CT2

X3 = CT2 + X3./CT3

X4 = CT3 + X4./(CT4 + CT5)

X10 = CTi_10 + CT11 + X10./CT10

X11 = CT10 + X11./CT11

X20 = CTi_20 + CT21 + X20./CT20

X21 = CT20 + X21./CT21

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

M1S = X2

M1D = X4

LA = X11

LV = X21

C.9. Exercício 9 – Tanques

Pretende-se implementar o comando do sistema apresentado na figura 64 que ilustra um sistema de enchimento/esvaziamento de dois tanques e tratamento do respetivo fluido, sendo que este último não é aqui descrito.

Este sistema é composto por:

- Dois tanques (tanque 1 e tanque 2).
- Sensores on-off que detetam se os tanques estão cheios ou vazios:
    - T1E e T1F, para tanque 1 vazio e tanque 1 cheio, respetivamente.
    - T2E e T2F, para tanque 2 vazio e tanque 2 cheio, respetivamente.
- Botão start, que serve para iniciar o funcionamento do sistema.
- Botão stop, que serve para parar o funcionamento sistema, no final do ciclo.
- Botão re_start, que serve para re-iniciar o funcionamento do sistema, após falha.
- V1 que é a válvula de enchimento do tanque 1.
- V2 que é, simultaneamente, a válvula de esvaziamento do tanque 1 e enchimento do tanque 2.
- V3 que é a válvula de esvaziamento do tanque 2.
- Arbitre e decida sobre todos os outros dados (sensores e atuadores) que, eventualmente, não estejam aqui indicados e que ache pertinente considerar.

O funcionamento é o seguinte:

- O enchimento do tanque 1 é feito através da abertura da válvula V1, quando o tanque estiver cheio a válvula V1 fecha. Depois de um tempo de espera de 10 segundos, a válvula V2 abre e o fluido para do tanque 1 para o tanque 2. Quando o tanque 1 estiver vazio, a válvula V2 fecha e, depois de um tempo de espera de 15 segundos, a válvula V3 abre e o fluido sai do tanque 2. Finalmente, quando o tanque 2 estiver vazio a válvula V3 fecha.
- Enquanto se encontra em processamento o liquido no tanque 1, também se encontra em processamento o liquido no tanque 2, daí que só se possa transferir o líquido do tanque 1 para o tanque 2 quando o tanque 2 esteja vazio.

- O funcionamento anteriormente descrito pode ser influenciado pelos botões start e stop.
- Se o tanque 1 demorar mais de 12 segundos a encher, significa que há um problema de funcionamento do sistema. Nesse caso, deve acender-se uma luz vermelha, que só se apagará quando o operador se dirigir ao equipamento – colocar manualmente o líquido que falta para que o tanque encha – e, finalmente carregar o botão re-start.
- Deve ser possível conhecer o volume de solução que passa pelo tanque 2, no final de cada dia de trabalho (consideres que o sistema funciona entre as 06:00h e 22:00h), tendo em conta que o tanque 2 é de 1,5 metros cúbicos.

Notas:

1. Todas as válvulas são monoestáveis; todos os sensores são normalmente abertos;
2. Há algumas operações executadas nos fluidos, em ambos os tanques, (misturas, aquecimentos, arrefecimentos, evaporações…) que não são aqui referidas, numa tentativa de facilitar o raciocínio para resolução do problema.

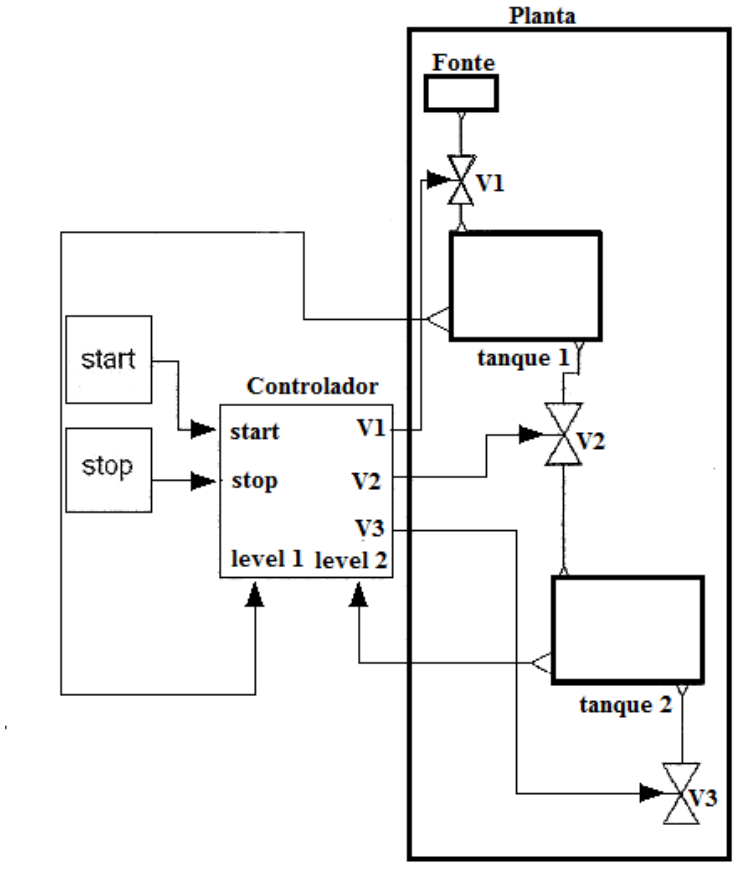


Figura 64 - Sistema de tanques automatizados e respetivos sensores e atuadores.

C.9.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 24 - Descrição, Words e Bits de todas as Entradas do Exercício 9.

Entrada	Descrição	Word & Bit
START	Botão de iniciação do sistema	0.00
STOP	Botão de paragem do sistema	0.01
RE_START	Botão de reinício do sistema	0.02
T1E	Tanque 1 vazio	0.03
T1F	Tanque 1 cheio	0.04
T2E	Tanque 2 vazio	0.05
T2F	Tanque 2 cheio	0.06

Tabela 25 - Descrição, Words e Bits de todas as Saídas do Exercício 9.

Saída	Descrição	Word & Bit
V1	Válvula 1	100.00
V2	Válvula 2	100.01
V3	Válvula 3	100.02
LV	Luz Vermelha	100.03
EL1	Início do turno de trabalho	100.04
EL2	Final do turno de trabalho	100.05

B - Especificação de Comando

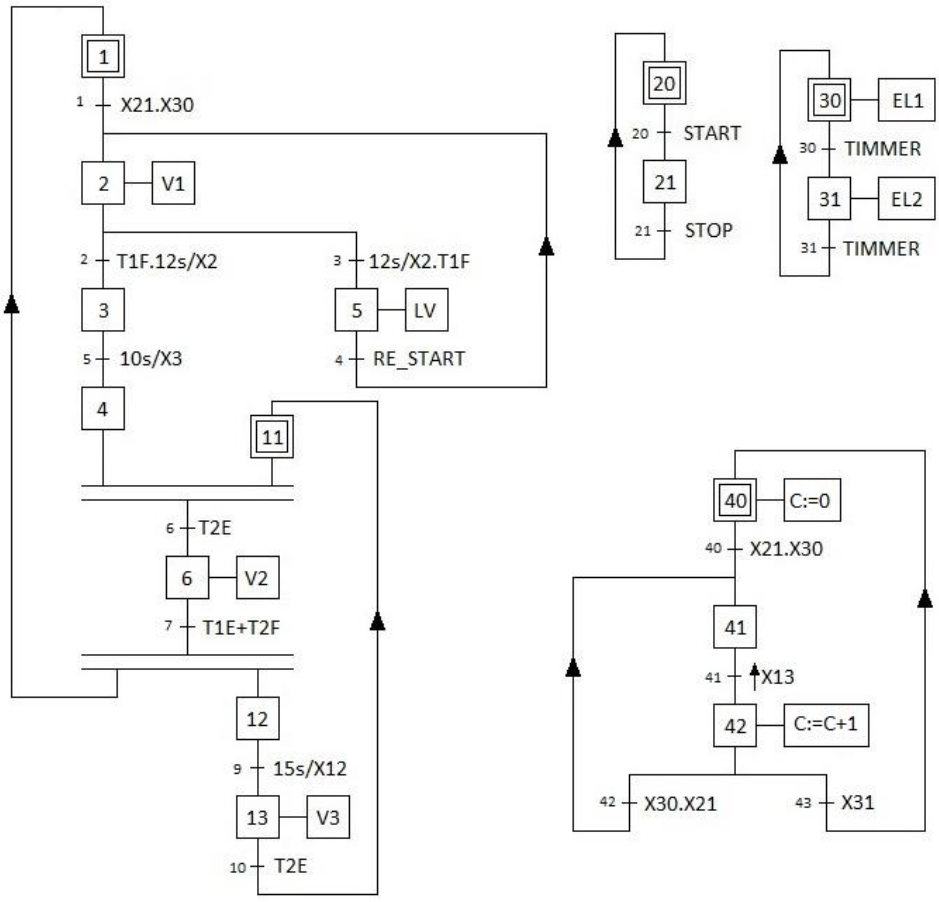


Figura 65 - Representação da modelização da parte de comando do exercício 9.

NOTA: por motivos académicos os Timmers DE CT30 e CT31 tem um tempo variável e por isso não têm um tempo especificado no Grafcet de comando.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_1=
/X1./X2./X3./X4./X5./X6./X7./X8./X9./X1
0./X11./X12./X13
CT1 = X1.X21.X30
CT2 = X2.T1F./(12s/X2)
CT3 = X2.12s/X2./T1F
CT4 = X5.RE_START
CT5 = X3.10s/X3
CT6 = X4.X11.T2E
CT7 = X6.(T1E + T2F)
CT8 = X12.15s/X12
CT9 =X13.T2E
CTi_20 = /X20./X21
CT20 = X20.START
CT21 = X21.STOP
CTi_30 =/X30./X31

CT30 = X30.TIMMER

CT31 = X31.TIMMER

CTi_40 = /X40./X41./X42

CT40 = X40.X21.X30

CT41 = X41.↑X13

CT42 = X42.X30.X21

CT43 =X42.X31

X21 = CT20+ X21./CT21

X30 = CTi_30 + CT31 + X30./CT30

X31 =CT30 + X31./CT31

X40 = CTI_40 + CT43 + X40./CT40

X41 = CT40 + CT42 + X41./CT41

X42 = CT41 + X42./(CT42 + CT43)

Em seguida elabora-se a atividade das etapas:

X1 = CTi_1 + CT7 + X1./CT1

X2 = CT1 + CT4 + X2./(CT2 + CT3)

X3 = CT2 + X3./CT5

X4 = CT5 + X4./CT6

X5 = CT3 + X5./CT4

X6 = CT6 + X6./CT7

X12 = CT7 + X12./CT8

X13 = CT8 + X13./CT9

X20 = CTi_20 + CT21 + X20./CT20

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

V1 = X2

V2 = X6

V3 = X13

LV = X5

EL1 = X30

EL2 = X31

C.10. Exercício 10 - Estação de Furação

Tendo em conta o sistema da figura 66, pretende-se elaborar a especificação do respetivo comando.

O início do ciclo dá-se premindo o botão Start e no início de funcionamento, todos os cilindros devem estar recuados e o motor elétrico deverá estar parado.

Somente é considerado um sensor SE1 no sistema para deteção de peças prontas para entrarem no sistema, de resto considera-se um funcionamento normal, onde há sempre peça. Devem ser associados dois fins-de-curso a cada um dos cilindros. Todos estes serão de simples efeito exceto o cilindro F. toas as electroválvulas de comando serão monoestáveis, exceto a electroválvula de comando do cilindro R que será biestável. O sistema possui três postos de trabalho e pretende-se que se realizem as três tarefas em simultânea:

- Carregamento de uma peça.
- Prisão e furação de uma peça.
- Teste e evacuação de uma peça.

O cilindro de rotação permite uma rotação de 120° do prato.

O teste é feito por um cilindro que desce até à sua posição inferior se o furo estiver efetuado. Se após um determinado tempo, o cilindro de teste não descer até à sua posição inferior, significa que o furo não foi corretamente efetuado e então a máquina deve parar até que alguém remova a peça manualmente e rearme a máquina premindo o botão RE_START.

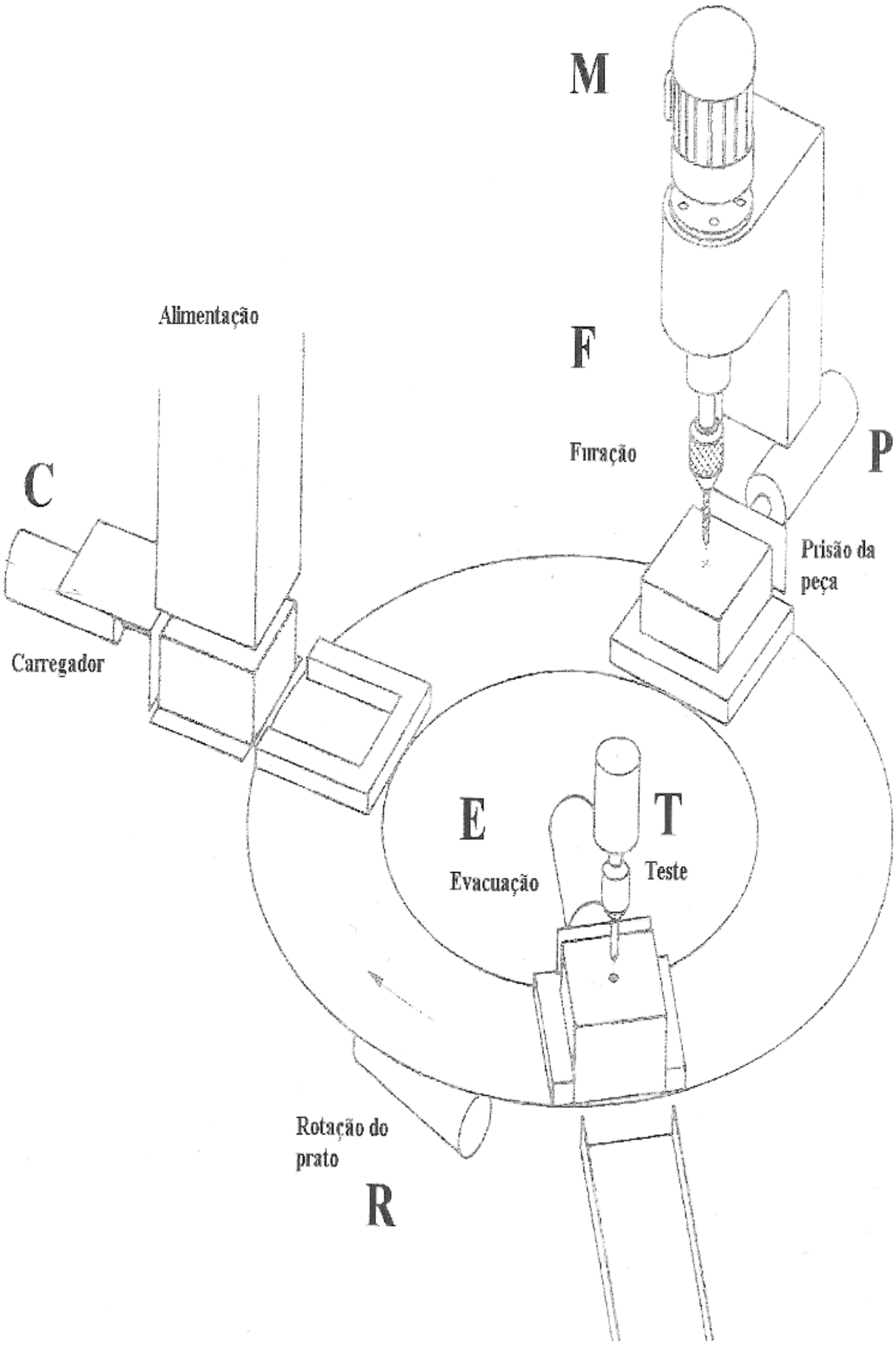


Figura 66 - Estação de Furação automatizada.

C.10.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 26 - Descrição, Words e Bits de todas as Entradas do Exercício 10.

Entrada	Descrição	Word & Bit
START	Botão de início de funcionamento do sistema	0.00
RE_START	Botão de reinício de funcionamento do sistema	0.01
S_C0	Cilindro C recuado	0.02
S_C1	Cilindro C avançado	0.03
S_P0	Cilindro P recuado	0.04
S_P1	Cilindro P avançado	0.05
S_F0	Cilindro F recuado	0.06
S_F1	Cilindro F avançado	0.07
S_T0	Cilindro T recuado	0.08
S_T1	Cilindro T avançado	0.09
S_E0	Cilindro E recuado	0.10
S_E1	Cilindro E avançado	0.11
S_R1	Cilindro R avançado	1.00
STOP	Botão de paragem do sistema	1.01
S_R0	Cilindro R recuado	1.02
SE1	Sensor de deteção de peças prontas para entrarem no sistema	1.03

Tabela 27 - Descrição, Words e Bits de todas as Saídas do Exercício 10

Saída	Descrição	Word & Bit
EVC	Ordem de avanço do cilindro C	100.00
EVF	Ordem de avanço do cilindro F	100.01

EVP	Ordem de avanço do cilindro C	100.02
M	Ordem de funcionamento do motor	100.03
EVR_AV	Ordem de avanço do cilindro R	100.04
EVR_REC	Ordem de recuo do cilindro R	100.05
EVT	Ordem de avanço do cilindro T	100.06
EVE	Ordem de avanço do cilindro E	100.07

B - Especificação de Comando

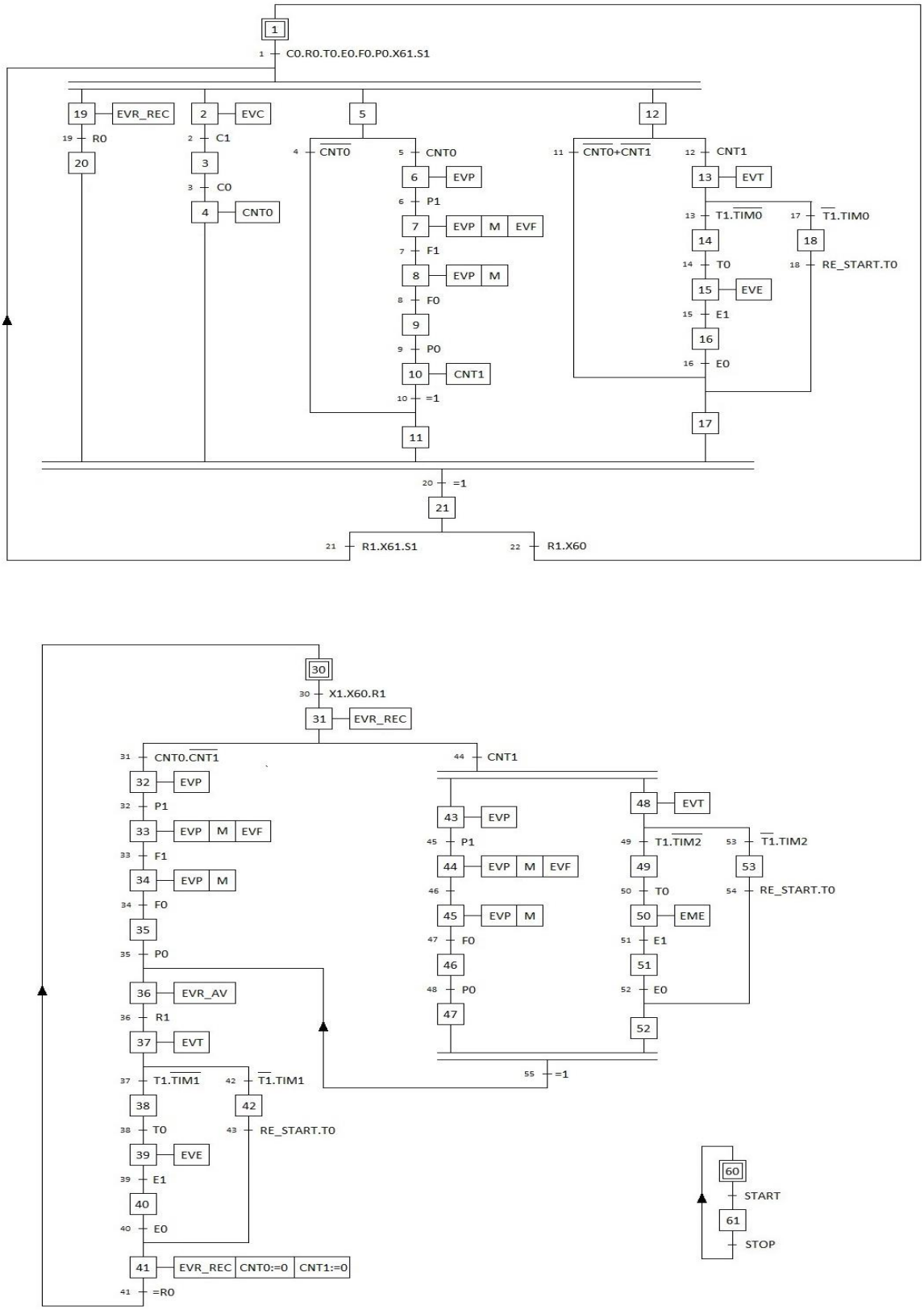


Figura 67 - Representação da modelação da parte de comando do exercício 10.

C - Conversão da especificação de comando para linguagem *Ladder*

Em função do SFC elaborado para o problema proposto (figura 67), converte-se o Grafcet para linguagem *Ladder*, como enunciado no subcapítulo 4.2.

As condições de transposição das etapas são:

CTi = /X1./X2./X3./X4./X5./X6./X7./X8./X9./X10./X11./X12./X13./X14./X15./X16./X17./X18./X19./X20./X21

CT1 = X1.S_C0.S_R0.S_T0.S_E0.S_F0.S_P0.X61.SE1

CT2 = X2.S_C1

CT3 = X3.S_C0

CT4 = X5./CNT0

CT5 = X5.CNT0

CT6 = X6.S_P1

CT7 =X7.S_F1

CT8 = X8.S_F0

CT9 =X9.S_P0

CT10 = X10

CT11 = X12./CNT0./CNT1

CT12= X12.CNT1

CT13 = X13.S_T1./TIM0

CT14 = X14.S_T0

CT15 = X15.S_E1

CT16 = X16.S_E0

CT17 = X13./S_T1.TIM0

CT18 = X18.RE_START.S_T0

CT19 =X19.S_R0

CT20 = X4.X11.X17.X20

CT21 = X21.S_R1.X61.SE1

CT22 = X21.S_R1.X60

CTi_30 =/X30/X31/X32/X33/X34/X35/X36/X37/X38/X39/X40/X41/X42/X43/X44X45/X46/X47/X48/X49/X50/X51/X52/X53

CT30 = X30.X1.X60.S_R1

CT31 = X31.CNT0./CNT1

CT32 = X32.S_P1

CT33 = X33.S_F1

CT34 = X34.S_F0

CT35 = X35.S_P0

CT36 = X36.S_R1

CT37 = X37.S_T1./TIM1

CT38 = X38.S_T0

CT39 = X39.S_E1

CT40 = X40. S_E0

CT41 = X41.S_R0

CT42 =X37./S_T1.TIM1

CT43 =X42.RE_START.S_T0

CT44 = X31.CNT1

CT45 = X43.S_P1

CT46 =X44.S_F1

CT47 = X45.S_F0

CT48 =X46.S_P0

CT49 = X48.S_T1./TIM2

CT50 =X49.S_T0

CT51 =X50.S_E1

CT52 =X51.S_E0

CT53 = X48./S_T1.TIM2

CT54 = X53.RE_START.S_T0

CT55 = X47.X52

CTi_60 = /X60./X61

CT60 = X60.START

CT61 =X61.STOP

Em seguida elabora-se a atividade das etapas:

X1 = CTi + CT22 + X1./CT1

X2 = CT1 + CT21 + X2./CT2

X3 = CT2 + X3./CT3

X4 = CT3 + X4./CT20

X5 = CT1 + CT21 + X5./(CT4 + CT5)

X6 = CT5 + X6./CT6

X7 = CT6 + X7./CT7

X8 = CT7 + X8./CT8

X9 = CT8 + X9./CT9

X10 = CT9 + X10./CT10

X11 = CT4 + CT10 + X11./CT20

X12 = CT1 + CT21 + X12./(CT11 + CT12)

X13 = CT12 + X13./(CT13 + CT14)

X14 = CT13 + X14./CT14

X15 = CT14 + X15./CT15

X16 =CT15 + X16./CT16

X17 =CT11 + CT16 + CT18 + X17./CT20

X18 = CT17 + X18./CT18

X19 = CT1 + CT21 + X19./CT19

X20 = CT19 + X20./CT20

X21 = CT20 + X21./(CT21 + CT22)

X30 = CTi_30 + CT41 + X30./CT30

X31 =CT30 + X31./(CT31 + CT44)

X32 = CT31 + X32./CT32

X33 = CT32 + X33./CT33

X34 = CT33 + X34./CT34

X35 = CT34 + X35./CT35

X36 =CT35 + CT55 + X36./CT36

X37 = CT36 + X37./(CT37 + CT42)

X38 = CT37 + X38./CT38

X39 = CT38 + X39./CT39

X40 = CT39 + X40./CT40

X41 = CT40 + CT43 + X41./CT41

X42 = CT42 + X42./CT43

X43 = CT44 + X43./CT45

X44 = CT45 + X44./CT46

X45 = CT46 + X45./CT47

X46 = CT47 + X46./CT48

X47 = CT48 + X47./CT55

X48 = CT44 + X48./(CT49 + CT53)

X49 = CT49 + X49./CT50

X50 = CT50 + X50./CT51

X51 =CT51 + X51./CT52

X52 = CT52 + CT54 + X52./CT55

X53 = CT53 + X53./CT54

X60 = CTi_60 + CT61 + X60./CT60

X61 = CT60 + X61./CT61

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

EVC = X2

EVF = X7 + X33 + X44

EVP = X6 + X7 + X8 + X32 + X33 + X34 + X43 + X44 + X45

M = X7 + X8 + X33 + X34 + X44 + X45

EVR_AV = X21 + X36

EVR_REC = X19 + X31 + X41

EVT = X13 + X37 + X48

EVE = X15 + X39 + X50

C.11. Exercício 11 – Estação de Transporte de Objetos

Tendo em conta o sistema da figura 68, pretende-se elaborar a especificação do respetivo comando.

A válvula V1 (monoestável) está 10 segundos aberta e 10 segundos fechada de modo a largar no tapete rolante um só objeto de cada vez. O sensor se1 ao detetar um objeto desencadeia a seguinte sequência: ativa a broca e o atuador para baixo; pára o atuador; fura durante 4 segundos; ativa o atuador para cima; pára a broca e abre V2 (monoestável). Em todos estes estados pode pressionar-se o botão B1 para parar a broca e o atuador. Se o sensor se2 detetar um objeto e o processo de furar não estiver concluído, então o tapete 1 pára, a válvula V1 fecha e fica à espera até o processo de furar acabar. Faça todas as considerações que achar conveniente. Identifique as entradas e saídas do controlador. Elabore o Grafcet nível 2.

Para além dos sensores atrás descritos, também existem dois sensores para determinar a posição da broca em cima ou em baixo, sendo estes sp0 e sp1 respetivamente.

Os tapetes possuem motores que os permitem rodar nas duas direções, dependendo da ordem emitida pelo controlador:

- M1 – Tapete superior roda para a frente.
- M1_REC – Tapete superior roda para trás.
- M2 – Tapete inferior roda para a frente.
- M2_REC – Tapete inferior roda para trás.

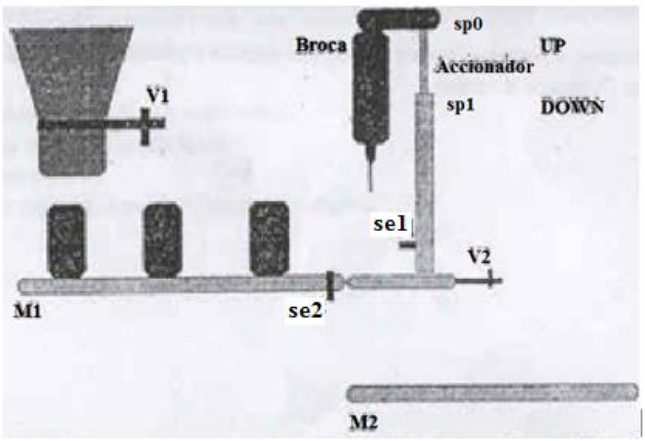


Figura 68 – Estação de transporte de peças e respetivos sensores e atuadores.

C.11.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 28 - Descrição, Words e Bits de todas as Entradas do Exercício 11.

Entrada	Descrição	Word & Bit
START	Botão de iniciação do sistema	0.00
se1	Sensor 1	0.01
se2	Sensor 2	0.02
sp0	Sensor de posicionamento da broca em cima	0.04
sp1	Sensor de posicionamento da broca em baixo	0.05
B1	Botão de paragem da broca e seu atuador	0.06
STOP	Botão de paragem do sistema	0.07

Tabela 29 - Descrição, Words e Bits de todas as Saídas do Exercício 11.

Saída	Descrição	Word & Bit
V1	Válvula 1 abre	100.00
M1	Tapete superior roda para a frente	100.01
A_Down	Atuador move-se para baixo	100.02
A_Up	Atuador move-se para cima	100.03

B	Acionamento do motor de rotação da broca	100.04
V2	Válvula 2 abre	100.05
M2	Tapete inferior roda para a frente	100.06
V1_F	Válvula 1 fecha	100.07
M1_REC	Tapete superior roda para trás	101.00
M2_REC	Tapete inferior roda para trás	101.01

B - Especificação de Comando

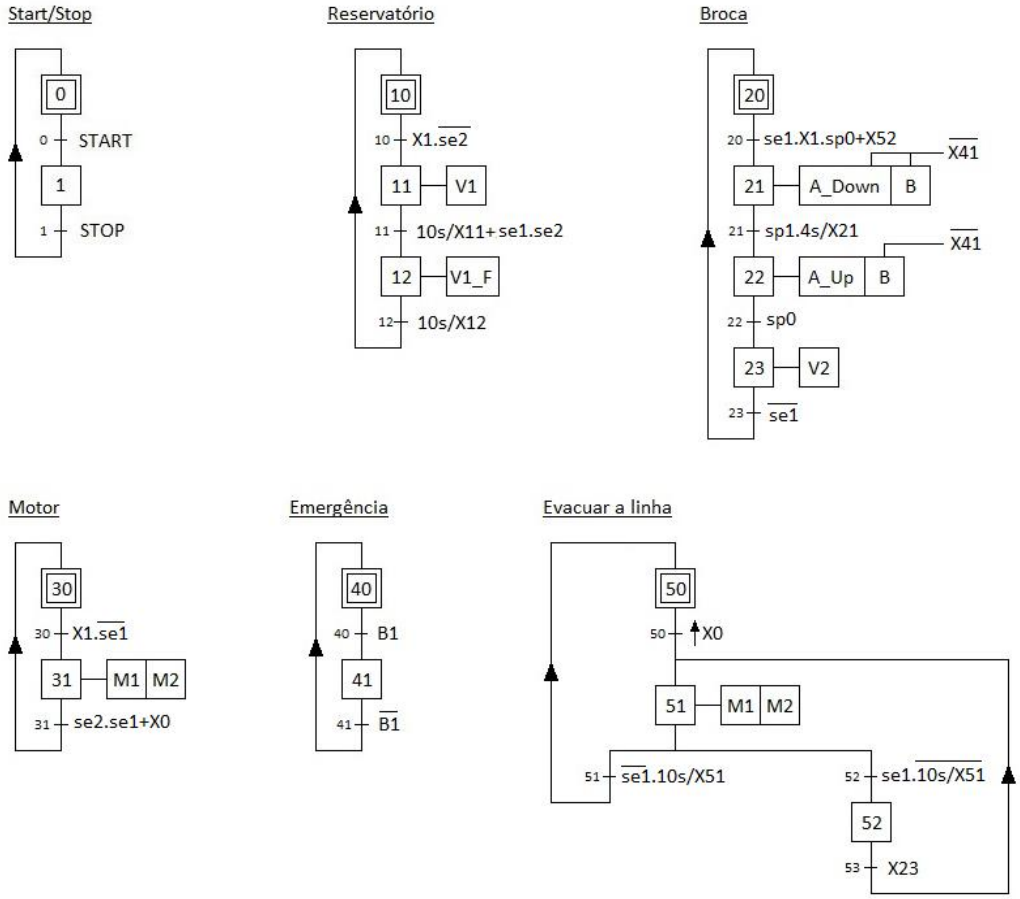


Figura 69 - Representação da modelização da parte de comando do exercício 11.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_0 = /X0./X1

CT0 = X0.START

CT1 = X1.STOP

CTi_10 = /X10./X11./X12

CT10 = X10.X1./se2

CT11 = X11.(10s/X11 + se1.se2)

CT12= X12.10s/X12

CT13 = X13.S_T1./TIM0

CTi_20 =/X20./X21./X22./X23

CT20 = X20.(se1.X1.sp0 + X52)

CT21 = X21.sp1.4s/X21

CT22 = X22.sp0

CT23 = X23./se1

CTi_30 =/X30./X31

CT30 = X30.X1./se1

CT31 = X31.(se2.se1 + X0)

CTi_40 = /X40./X41

CT40 = X40. B1

CT41 = X41./B1

CTi_50 = /X50./X51./X52

CT50 =X50.↑X0

CT51 =X51./se1.10s/X51

CT52 =X51.se1./(10s/X51)

CT53 = X52.X23

Em seguida elabora-se a atividade das etapas:

X0 = CTi_0 + CT1 + X0./CT0

X1 = CT0+ X1./CT1

X10 = CTi_10 + CT12 + X10./CT10

X11 = CT10 + X11./CT11

X12 = CT11 + X12./ CT12

X20 = CTi_20 + CT23 + X20./CT20

X21 = CT20 + X21./CT21

X22 = CT21 + X22./CT22

X23 =CT22 + X23./CT23

X30 = CTi_30 + CT31 + X30./CT30

X31 =CT30 + X31./CT31

X40 = CTi_40 + CT41 + X40./CT40

X41 = CT40 + CT43 + X41./CT41

X50 = CT50 + CT51 + X50./CT50

X51 =CT50 + CT53 + X51./(CT51 + CT52)

X52 = CT52 + X52./CT53

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

V1 = X11

V1_F = X12

V2 = X23

M1 = X31 + X51

M2 = X31 + X51

A_DOWN = X21./X41

A_UP = X22./X41

B = (X21 + X22)./X41

C.12. Exercício 12 – Grua

Tendo em conta o sistema da figura 70, pretende-se elaborar a especificação do respetivo comando.

O processo inicia-se com a grua no lado esquerdo (sensor se1) e com o cabo içado (sensor se4). O cabo deve descer através do motor M2 até ativar o sensor se5 e uma vez este ativado o motor M2 pára e fica em espera durante 10 segundos que é o tempo necessário para que o contentor seja colocado no estrado.

Neste intervalo de tempo pode-se pressionar um botão B1 cuja finalidade é manter o processo neste estado independentemente dos 10 segundos de espera.

Findo este tempo ou logo que o B1 transite de ON para OFF, o cabo deve subir até ativar o sensor se4; aqui o cabo pára de subir e o motor Ml (Right) é ativado para que a grua se desloque para a direita até atingir o sensor se2.

Uma vez neste estado o motor M1 pára e arranca o motor M2 (Down) de modo a que o cabo desça até atingir o sensor se5: Aqui deverá ser ativado o atuador durante dois segundos, mas com a condição de não existir nenhum contentor no tapete rolante. Caso haja, a grua deve esperar que o processo em curso termine e só depois é que o contentor deve ser co1ocado no tapete.

Após esta ação a grua deverá içar o cabo e voltar à posição inicial para retomar o processo. O tapete rolante, que está em movimento transporta o objeto até este ser detetado pelo sensor se3. Aí, o tapete pára e a válvula Vl abre durante 5 segundos. Findo este tempo a válvula fecha e o tapete volta a rolar: 2 segundos depois de o tapete estar em movimento deve parar novamente durante 4 segundos, de modo a que a carga seja removida, e no fim deste tempo o tapete volta outra vez a mover-se.

O sensor se3 deve contar os contentores que vão passando no tapete, e ao quarto contentor a válvula V2 é aberta durante 10 segundos.

Para além dos sensores atrás descritos existem também mais dois sensores, se6 e se7, que traduzem a posição da grua quando está a 90º para trás e para a frente respetivamente. Projete o controlador lógico.

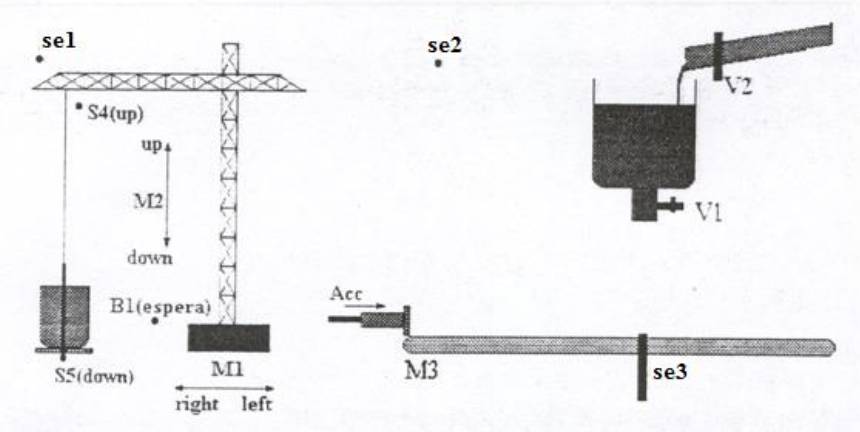


Figura 70 - Estação de transporte de peças com grua incorporada e respetivos sensores e atuadores.

C.12.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 30 - Descrição, Words e Bits de todas as Entradas do Exercício 12.

Entrada	Descrição	Word & Bit
START	Botão de iniciação do sistema	0.00
STOP	Botão de paragem do sistema	0.01
B1	Botão de paragem da grua no ponto de carga	0.02
se1	Sensor de posição da grua à esquerda	0.03
se2	Sensor de posição da grua à direita	0.04
se3	Sensor de posição da peça debaixo da válvula V1	0.05
se4	Sensor de posição da grua em cima	0.06
se5	Sensor de posição da grua em baixo	0.07

se6	Sensor de posição da grua 90º para trás	1.00
se7	Sensor de posição da grua 90º para a frente	1.01

Tabela 31 - Descrição, Words e Bits de todas as Saídas do Exercício 12.

Saída	Descrição	Word & Bit
M1_RIGHT	Grua move-se para a Direita	100.00
M1_LEFT	Grua move-se para a Esquerda	100.01
M2_DOWN	Grua move-se para baixo	100.02
M2_UP	Grua move-se para cima	100.03
M3	Tapete avança	100.04
AV	Atuador cilíndrico tira peça da grua	100.05
V1	Válvula 1 abre	100.06
V2	Válvula 2 abre	100.07
M3_REC	Tapete recua	100.08

B - Especificação de Comando

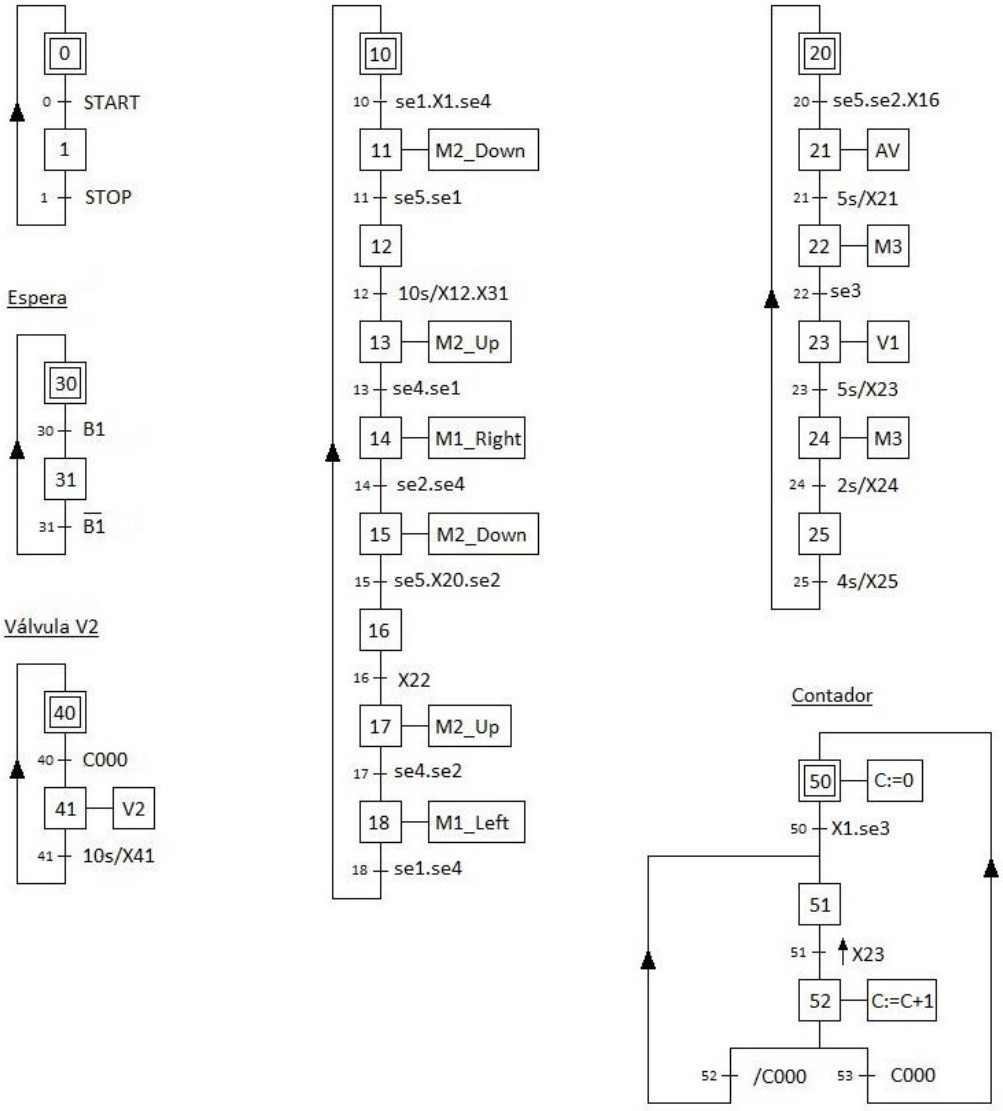


Figura 71 - Representação da modelização da parte de comando do exercício 12.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_0 = /X0./X1

CT0 = X0. START

CT1 = X1.STOP

CTi_10 =

/X10./X11./X12./X13./X14./X15./X16./X17./X18

CT10 = X10.s1.X1.se4

CT11 = X11.se5.se1

CT12= X12.10s/X12./X31

CT13 = X13.se4.se1

CT14 = X14.se2.se4

CT15 = X15.se5.X20.se2

CT16 = X16.X22

CT17 = X17.se4.se2

CT18 = X18.se1.se4

CTi_20 = /X20./X21./X22./X23./X24./X25

CT20 = X20.X16.se5.se2

CT21 = X21.5s/X21

CT22 = X22.se3

CT23 = X23.5s/X23

CT24 = X24.2s/X24

CT25 = X25.4s/X25

CTi_30 =/X30./X31

CT30 = X30.B1

CT31 = X31./B1

CTi_40 = /X40./X41

CT40 = X40.C000

CT41 = X41.10s/X41

CTi_50 = /X50./X51./X52

CT50 =X50.X1.se3

CT51 =X51.↑X23

CT52 =X52./C000

CT53 = X52.C000

Em seguida elabora-se a atividade das etapas:

X0 = CTi_0 + CT1 +X0./CT0

X1 = CT0 + X1./CT1

X10 = CTi_10 + CT18 + X10./CT10

X11 = CT10 + X11./CT11

X12 = CT11 X12./CT12

X13 = CT12 + X13./CT13

X14 = CT13 + X14./CT14

X15 = CT14 + X15./CT15

X16 =CT15 + X16./CT16

X17 = CT16 + X17./CT17

X18 = CT17 + X18./CT18

X20 = CTi_20 + CT25 + X20./CT20

X21 = CT20 + X21./CT21

X22 = CT21 + X22./CT22

X23 = CT22 + X23./CT23

X24 = CT23 + X24./CT24

X25 = CT24 + X25./CT25

X30 = CTi_30 + CT31 + X30./CT30

X31 =CT30 + X31./CT31

X40 = CTi_40 + CT41 + X40./CT40

X41 = CT40 + X41./CT41

X50 = CTi_50 + CT53 + X50./CT50

X51 =CT50 + CT52 + X51./CT51

X52 = CT51 + X52./(CT52 + CT53)

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

M1_RIGHT = X14

M1_LEFT = X18

M2_DOWN = X11 + X15

M2_UP = X13 + X17

M3_AV = X22 + X24

AV = X21

V1 = X23

V2 = X41

C.13. Exercício 13 – Garra e Carros de transporte

Na figura 72 encontra-se representado o sistema automatizado e modelar. Os carros C1 e C2 transportam peças entre os postos A1 e A2 e o posto B, partilhando o mesmo sistema de descarga comum em B. O movimento dos carros é o seguinte: com os dois carros nos postos de repouso (A1 e A2), após a ordem de partida (START), é dada a ordem AV2 ao carro C2, deslocando-se até B onde é descarregado. Terminada a operação de descarga, é dada a ordem REC2 para o carro C2 regressar ao posto A2. De seguida, é dada a ordem AV1 ao carro C1 para se deslocar até ao posto B onde é por sua vez, é descarregado. Após a ordem REC1, regressa ao posto A1. O sistema de descarga é constituído por uma pinça que pode subir (SP) e descer (DP), existindo dois sensores nas posições superior (se5) e inferior (se6). Pode ainda rodar à direita (RD) e rodar à esquerda (RE), podendo ser detetada a sua posição em cada 90º através dos sensores de posicionamento se1, se2, se3 e se4. O fecho da pinça (FP) é comandado por um êmbolo de efeito simples, sendo detetado pelo contacto PF. O ciclo de operações envolvido na descarga é o seguinte:

- Inicialmente a pinça está na posição superior direita e aberta.
- Após ser solicitada a descarga de um carro, a pinça desce, agarra o objeto, sobe, roda à esquerda e abre depositando o objeto no tapete de evacuação.

De seguida, regressa à posição inicial. Para ativar o tapete de evacuação é dada a ordem M. Cada carro possui um sensor que tem como função detetar quando se encontra um objeto no seu interior (fc1 para o C1 e fc2 para C2).

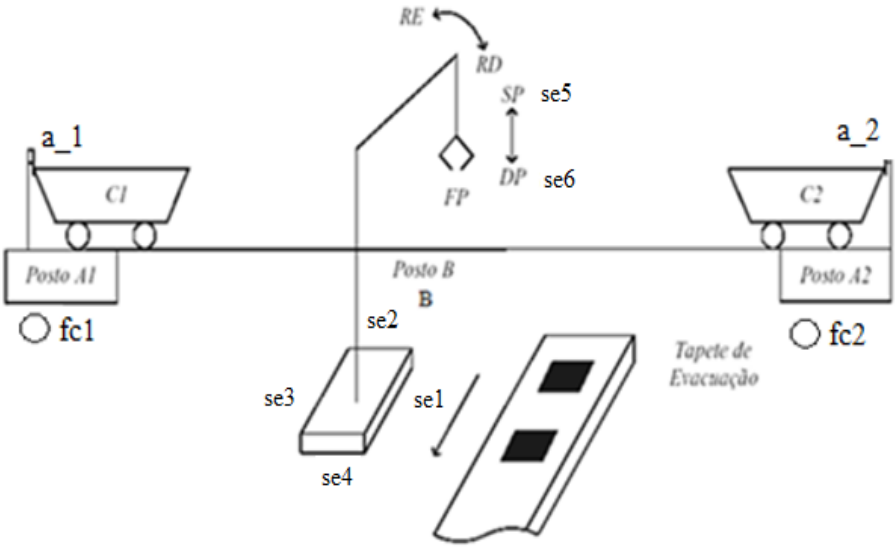


Figura 72 - Sistema automatizado a modelar.

C.13.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 32 - Descrição, Words e Bits de todas as Entradas do Exercício 13.

Entrada	Descrição	Word & Bit
START	Botão de iniciação do sistema	0.00
STOP	Botão de paragem do sistema	0.01
A_1	Carro na posição A_1	0.02
A_2	Carro na posição A_2	0.03
B	Carro no posto B	0.04
pf	Pinça fechada	0.05
fc1	Carro 1 carregado	0.06
fc2	Carro 2 carregado	0.07
se1	Pinça na posição inicial	0.08
se2	Pinça sobre o posto B	0.09
se3	Pinça à esquerda	0.10
se4	Pinça para a frente	0.11
se5	Pinça em cima	1.00
se6	Pinça em baixo	1.01

Tabela 33 - Descrição, Words e Bits de todas as Saídas do Exercício 13.

Saída	Descrição	Word & Bit
SP	Pinça sobe	100.00
DP	Pinça desce	100.01
FP	Pinça fecha	100.02
RE	Pinça roda à esquerda	100.03
RD	Pinça roda à direita	100.04
AV1	Carro 1 avança	100.05
REC1	Carro 1 recua	100.06
AV2	Carro 2 avança	100.07
REC2	Carro 2 recua	101.00
M1	Tapete de evacuação move-se	101.01

B - Especificação de Comando

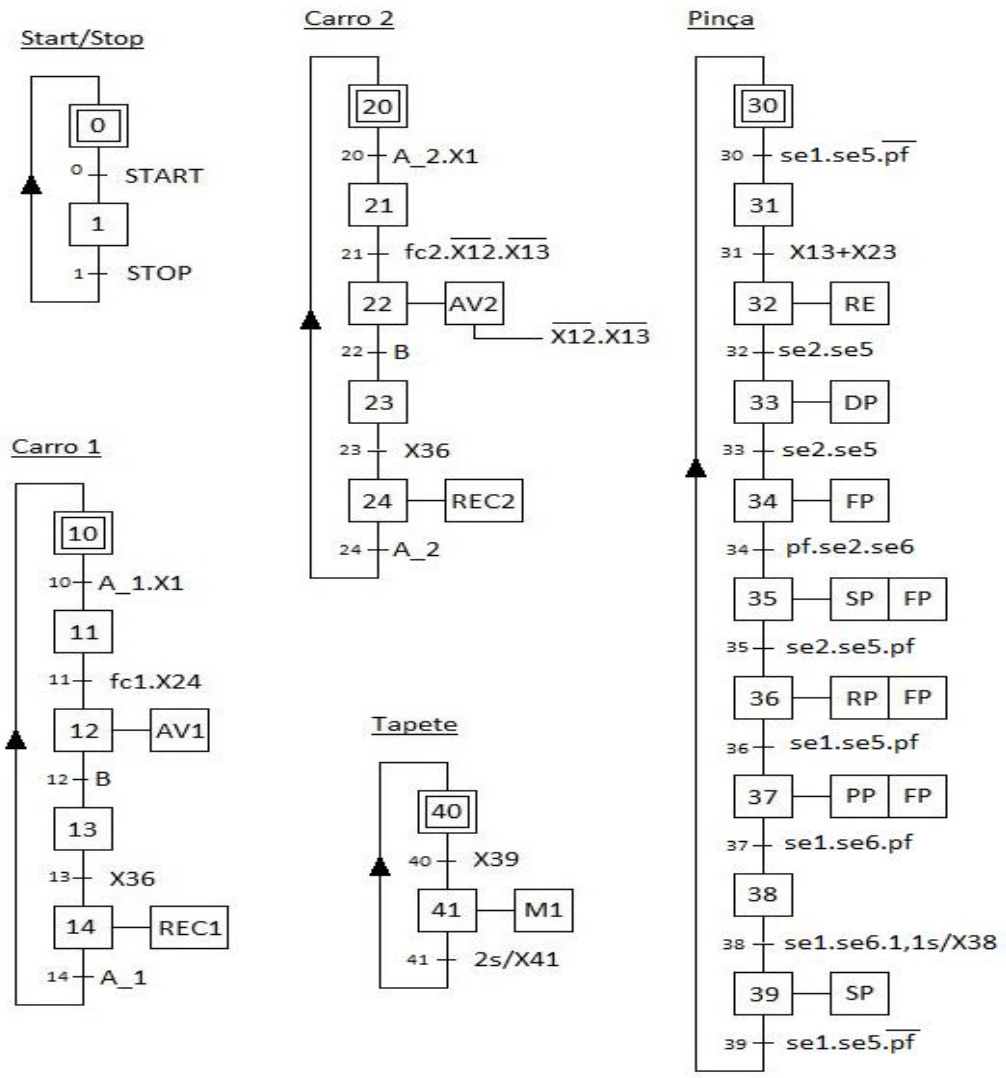


Figura 73 - Representação da modelização da parte de comando do exercício 13.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_0 = /X0./X1

CT0 = X0.START

CT1 = X1.STOP

CTi_10 =/X10./X11./X12./X13./X14

CT10 = X10.A_1.X1

CT11 = X11.fc1.X24

CT12= X12.B

CT13 = X13.X36

CT14 = X14.A_1

CT15 = X15.S_E1

CTi_20 = /X20./X21./X22./X23./X24

CT20 = X20.A_1.X1

CT21 = X21.fc2./X12./X13

CT22 = X22.B

CT23 = X23.X36

CT24 = X24.A_2

CTi_30

=/X30./X31./X32./X33./X34./X35./X36./X37./X38./X39

CT30 = X30.se1.se5./pf

CT31 = X31.(X13 + X23)

CT32 = X32.se2.se5

CT33 = X33.se2.se6

CT34 = X34.pf.se2.se6

CT35 = X35.pf.se2.se5

CT36 = X36.pf.se1.se5

CT37 = X37.pf.se1.se6

CT38 = X38./pf.se1.se6. 1,1s/X38

CT39 = X39./pf.se1.se5

CTi_40 = /X40./X41

CT40 = X40. X39

CT41 = X41.2s/X41

Em seguida elabora-se a atividade das etapas:

X0 = CTi_0 + CT1 + X0./CT0

X1 = CT0 + X1./CT1

X10 = CTi_10 + CT14 + X10./CT10

X11 = CT10 + X11./CT11

X12 = CT11 + X12./CT12

X13 = CT12 + X13./CT13

X14 = CT13 + X14./CT14

X20 = CTi_20 + CT24 + X20./CT20

X21 = CT20 + X21./CT21

X22 = CT21 + X22./CT22

X23 = CT22 + X23./CT23

X24 = CT23 + X24./CT24

X30 = CTi_30 + CT39 + X30./CT30

X31 =CT30 + X31./CT31

X32 = CT31 + X32./CT32

X33 = CT32 + X33./CT33

X34 = CT33 + X34./CT34

X35 = CT34 + X35./CT35

X36 =CT35 + X36./CT36

X37 = CT36 + X37./CT37

X38 = CT37 + X38./CT38

X39 = CT38 + X39./CT39

X40 = CTi_40 + CT41 + X40./CT40

X41 = CT40 + X41./CT41

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

SP = X35 + X39

DP = X33 + X37

FP = X34 + X35 + X36 + X37

RE = X32

RD = X36

AV1 = X12

REC1 = X14

AV2 = X22 ./X12 ./X13

REC2 = X24

M1 = X4

C.14. Exercício 14 – Carros de transporte

Apresente, utilizando o formalismo SFC da norma IEC 60848, a especificação para o sistema apresentado na figura 74.

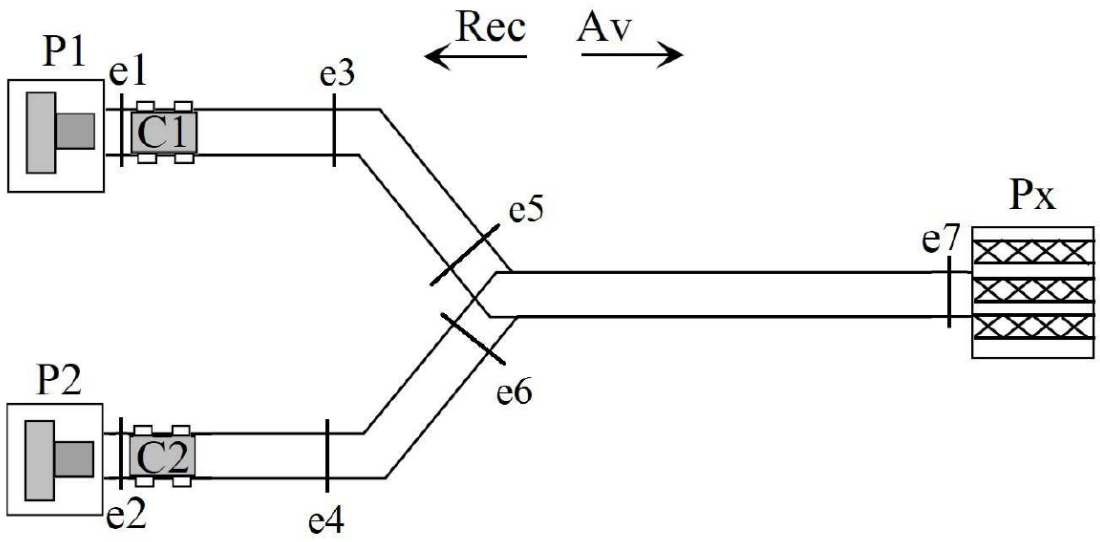


Figura 74 - Sistema automatizado a modelizar

Na figura está representado o sistema de transporte de materiais entre dois postos de trabalho P1 e P2 e um armazém Px. O carro C1 efetua o transporte de material entre o posto de fabrico P1 e o posto de entrada do armazém Px. O movimento deste carro é comandado pelos sinais AV1 e REC1, respetivamente, para fazer avançar C1 no sentido de P1 para Px, e para fazer C1 recuar de Px para P1.

- Uma vez no posto P1 (e1 atuado), aguarda a ativação do sinal fc1 que indica o fim da operação de carga neste posto. Logo que esteja carregado, deve deslocar-se no sentido do avanço.
- Uma vez chegado a Px (e7 atuado) aguarda o sinal fdx que indica a conclusão das operações de descarga em Px, após o que regressa a P1.

De forma idêntica a C1, o carro C2, que transporta materiais entre P2 e Px, é comandado pelos sinais AV2 e REC2, entre e2 e e7, sendo que a ativação do sinal fc2 indica o fim de operação de carga no posto P2 e ativação do sinal fdx indica a conclusão das operações de descarga em Px.

Na especificação do comando dos carros C1 e C2 deve ter ainda em conta o seguinte:

- Se C1 estiver no troço e5-e7, o carro C2 não poderá estar no troço e6-e7 e vice-versa.
- Se o carro C1 estiver no troço e5-e7 ou se o carro C2 estiver no troço e6-e7 deve ligar-se uma luz amarela que deve desligar-se logo que tal situação não se verifique.
- Por questões de segurança, os troços e3-e5 e e4-e6 não poderão ter carro em simultâneo, no sentido de avanço.
- No caso dos sensores e3 e e4 serem atuados no mesmo instante, quando os carros vão no sentido de avanço, deve ser dada prioridade de passagem ao carro C1.
- Considere dois botões, start e stop, para colocar o sistema a funcionar/parar. Na paragem, o sistema deve sempre parar na posição ilustrada na figura 74.
- Após períodos de 1500 ciclos, C2 deve parar para manutenção. Conte o número de cargas que faz o carro C2.
- Considere como posição inicial a que está ilustrada na figura 74.
- Considere que pretende fazer o máximo de ciclos possível por dia.
- Considere sensores e botoneiras normalmente abertos.

C.14.1. Modelo do Controlador

A - Variáveis de Entrada e Saída

Tabela 34 - Descrição, Words e Bits de todas as Entradas do Exercício 14.

Entradas	Descrição	Word & Bit
start	Botão de inicialização do sistema	0.00
stop	Botão de paragem do sistema	0.01
sensor_e1	Carro 1 na posição e1 (P1)	0.02
sensor_e2	Carro 2 na posição e2 (P2)	0.03
sensor_e3	Carro1 na posição e3	0.04

sensor_e4	Carro 2 na posição e4	0.05
sensor_e5	Carro 1 na posição e5	0.06
sensor_e6	Carro 2 na posição e6	0.07
sensor_e7	Carro na posição e7 (Px)	0.08
fc1	Carro 1 carregado	0.09
fc2	Carro 2 carregado	0.10
fdx	Carro descarregado	0.11

Tabela 35 - Descrição, Words e Bits de todas as Saídas do Exercício 14.

Saídas	**Descrição**	**Word & Bit**
AV1	Carro 1 avança	100.00
AV2	Carro 2 avança	100.01
REC1	Carro 1 recua	100.02
REC2	Carro 2 recua	100.03
LA	Luz amarela acende-se	100.04

B - Especificação de Comando

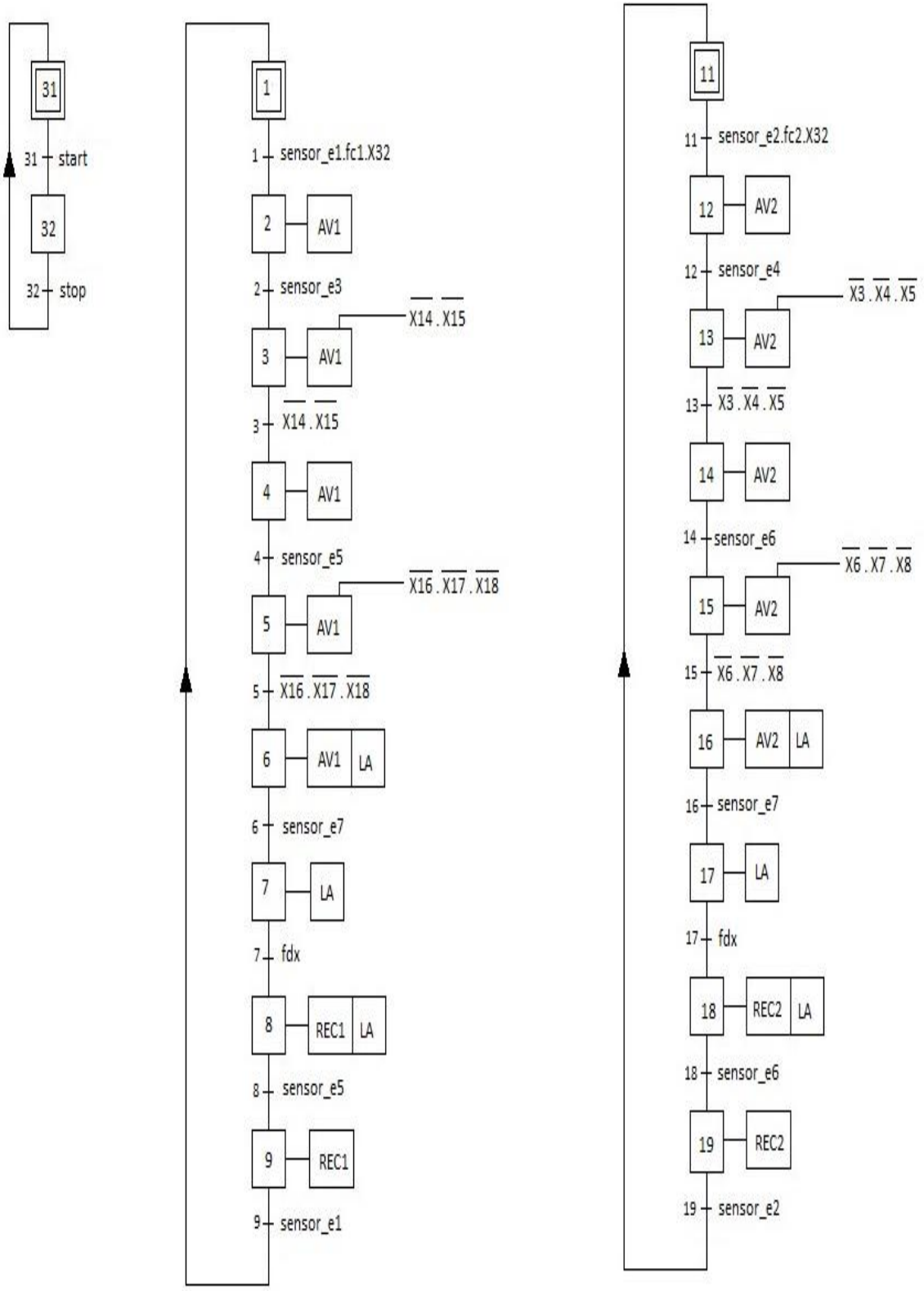


Figura 75 - Representação da modelização da parte de comando do exercício 14.

C - Conversão da especificação de comando para linguagem *Ladder*

CTi_1=
/X0./X1./X2./X3./X4./X5./X6./X7./X8.
/X9

CT1 = X1.sensor_e1.fc1.X32
CT2 = X2. sensor_e3
CT3 = X3./X14./X15
CT4 = X4. sensor_e5
CT5 = X5./X16./X17./X18
CT6 = X6. sensor_e7
CT7 = X7.fdx
CT8 = X8.sensor_e5
CT9 = X9. sensor_e1
CTi_2 =
/X11./X12./X13./X14./X15./X16./X17.
/X18./X19
CT11 = X11.sensor_e2.fc1.X32
CT12 = X12. sensor_e4
CT13 = X13./X3./X4./X5
CT14 = X14. sensor_e6
CT15 = X15./X6./X7./X8
CT16 = X16. sensor_e7
CT17 = X17.fdx
CT18 = X18.sensor_e6
CT19 = X19. sensor_e2
CTi_3 = /X31./X32
CT31 = X31.start
CT32 = X32.stop

Em seguida elabora-se a atividade das etapas:
X1 = CTi_1 + CT9 + X1./CT1
X2 = CT1 + X2./CT2
X3 = CT2 + X3./CT3
X4 = CT3 + X4./CT4
X5 = CT4 + X5./CT5
X6 = CT5 + X6./CT6
X7 = CT6 + X7./CT7
X8 = CT7 + X8./CT8
X9 = CT8 + X9./CT9
X11 = CTi_2 + CT19 + X11./CT11
X12 = CT11 + X12./CT12
X13 = CT12 + X13./CT13
X14 = CT13 + X14./CT14
X15 = CT14 + X15./CT15
X16 = CT15 + X16./CT16
X17 = CT16 + X17./CT17
X18 = CT17 + X18./CT18
X19 = CT18 + X19./CT19
X31 = CTi_3 + CT32 + X31./CT31
X32 = CT31 + X32./CT32

Para finalizar, elabora-se os pontos que cada etapa ativa/desativa.

AV1 = X2 + X3./X14./X15 + X4 + X5./X16./X17./X18 + X6

REC1 = X8 + X9

AV2 = X12 + X13./X3./X4./X5 + X14 + X15./X6./X7./X8 + X16

REC2 = X18 + X19

LA = X6 + X7 + X8 + X16 + X17 + X18