Instituto Superior de
Engenharia de Coimbra

Departamento

de Engenharia Eletrotécnica

Desenvolvimento de aplicações para a indústria cerâmica usando PLC Siemens

Relatório de Estágio apresentado para a obtenção do grau de Mestre em Engenharia Eletrotécnica – Área de Especialização em Automação e Comunicações em Sistemas de Energia

Autor

André Saraiva Correia

Orientadores

João Paulo Morais Ferreira

Frederico Miguel do Céu Marques dos Santos

Instituto Superior de Engenharia de Coimbra

Supervisor na Empresa

Engenheiro André Henriques Simões Seabra da Costa

SA – Soluções em Automação, S.A.

Coimbra, junho, 2015

Instituto Superior de
Engenharia de Coimbra

Departamento

de Engenharia Eletrotécnica

Desenvolvimento de aplicações para a indústria cerâmica usando PLC Siemens

Relatório de Estágio apresentado para a obtenção do grau de Mestre em Engenharia Eletrotécnica – Área de Especialização em Automação e Comunicações em Sistemas de Energia

Autor

André Saraiva Correia

Orientadores

João Paulo Morais Ferreira

Frederico Miguel do Céu Marques dos Santos

Instituto Superior de Engenharia de Coimbra

Supervisor na Empresa

Engenheiro André Henriques Simões Seabra da Costa

SA – Soluções em Automação, S.A.

Coimbra, junho, 2015

AGRADECIMENTOS

Durante a realização do estágio, queria agradecer ao meu tutor do estágio Eng.º André Henrique Simões Seabra da Costa, à administração da empresa SA – Soluções em Automação, S.A., aos meus orientadores do Instituto Superior de Engenharia de Coimbra, João Paulo Ferreira e Frederico Miguel dos Santos pela orientação, dedicação e conselhos dados durante o estágio. Também queria agradecer ao meu amigo Carlos Filipe Saraiva que me acompanhou durante o estágio e me ajudou sempre que precisei.

Também quero agradecer aos meus colegas de trabalho que me apoiaram na elaboração deste trabalho e à minha família que sempre demonstrou todo o apoio que precisei.

Agradeço também, aos vários amigos que me ajudaram, não só com o esclarecimento de determinadas dúvidas mas também pelos incentivos que me dispensaram.

RESUMO

O presente Relatório enquadra-se no âmbito da Unidade Curricular de Estágio do Mestrado em Engenharia Eletrotécnica, ramo de Automação e Comunicações em Sistemas de Energia, do Instituto Superior de Engenharia de Coimbra subordinado ao tema "Desenvolvimento de aplicações para a indústria cerâmica usando PLC Siemens".

O estágio foi realizado na empresa SA – Soluções em Automação, S.A., empresa sediada em Oliveira do Bairro, com vasta experiência na realização de projetos de automação para a indústria cerâmica.

Durante a realização do estágio foi dada a oportunidade ao estagiário de contribuir ativamente para o desenvolvimento e aperfeiçoamento de aplicações para controlo de máquinas da indústria cerâmica, inserindo-se numa equipa de trabalho com vasta experiência na área.

As aplicações desenvolvidas aplicam-se a quase todo o processo de fabrico de uma fábrica de cerâmica estrutural (tijolos), proporcionando ao estagiário um conhecimento mais íntimo deste tipo de indústria e os seus desafios.

As funções foram desenvolvidas com vista a serem utilizadas em autómatos da marca *Siemens*, o que se tornou uma mais-valia, tornando possível adquirir conhecimentos de programação e parametrização de autómatos de uma das marcas líderes de mercado e com grande utilização num vasto número de indústrias para além da cerâmica.

Muitas das máquinas para as quais se criaram as aplicações possuem acionamentos mecânicos que utilizam variadores eletrónicos de velocidade (VEV's), esta utilização possibilitou a tomada de conhecimento e criação de aplicações próprias para comunicação com VEV's da marca SEW *Eurodrive*, marca utilizada na empresa onde se realizou o estágio. Esta necessidade de aprender a trabalhar com este tipo de equipamentos revelou-se outra mais-valia para o estagiário possibilitando a interação, aprendizagem e controlo de equipamentos da marca SEW.

Após a fase inicial do estágio em que foi dada formação em PLC's Siemens, foram desenvolvidas funções para o controlo de máquinas para a indústria cerâmica, tais como laminadores, transportadores, ventilação interna de secadores semi-contínuos, queimadores pulsados, etc. Terminadas as funções foram testadas dentro do possível, tentando replicar as condições de funcionamento reais das máquinas às quais as funções se aplicavam. Durante os testes foram descobertos alguns erros no desenvolvimento dos programas, que após identificados foram prontamente resolvidos.

Devido ao fato de não existir nenhuma fábrica em fase de programação e arranque durante a realização do estágio, não foi possível testar o trabalho realizado no estágio em situação de funcionamento real das máquinas. Este aspeto fez com que as funções não pudessem ter sido completamente testadas. Apenas alguns meses depois de terminado o estágio é que as funções serão colocadas em funcionamento numa fábrica nova.

ABSTRACT

The aim of the present report entitled "Development of applications for the ceramic industry using Siemens PLC" is to approach an internship related with an Electrical Engineering Master's Degree specialized in Automation and Communications in Energy Systems at the Engineering Institute of Coimbra.

The related internship was held in SA – Soluções em Automação, S.A., a company with a vast knowledge in automation projects execution for the ceramics industry based in Oliveira do Bairro.

During the internship, included in a team with a lot of experience in this area, the trainee had the opportunity to develop and improve applications to control machines of the ceramics industry.

The developed applications could be applied in the entire manufacturing process of a ceramic industry, and this is because why the trainee had the chance to face different challenges and acquire a varied knowledge in the area.

In order to use PLCs from Siemens, the developed work had result as an additional value for the trainee once it was possible for him to acquire knowledge in PLCs programming of one of the industry leading brands in industry, including the ceramics.

As the machines whose applications were made for have mechanical drives with electronic speed drivers (VEV's), it was required knowledge to create applications able to communicate with SEW Eurodrive VEV's. This fact enabled the control and interaction of SEW´s equipment's.

The first stage of the internship was related with Siemens PLC´s training, which allowed the development of ceramics industry machinery applications such as, mills, conveyors, ventilation of semi-continuous dryer, pulsed burners, etc. After the mentioned training completion, the developed applications were tested taking into account real operating conditions.

Because there was no factory in programming and performance stage during the internship period, it was no possible to test the same applications in real situations. This circumstance meant that functions were not fully tested during the internship and this will only happen after the internship completion in a new fabric.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIMBOLOGIA

CCW – *Counter Clock Wise*

CW – *Clock Wise*

DB – *Data Block*

FB – *Function Block*

FC – *Function*

I/O – *Imput/Output*

OB – *Organization Block*

PID – Controlador Proporcional, Integral e Derivativo

SFB – *System Function Block*

SFC – *System Function*

STL – *Statement List*

ABREVIATURAS

CPU – *Central Processing Unit*

HMI – *Human Machine Interface*

PLC – *Programmable Logic Controller*

SCADA – *Supervisory Control and Data Acquisition*

1. INTRODUÇÃO

1.1. Contextualização do Estágio

O presente Relatório de Estágio surge no âmbito da unidade curricular de Estágio do Mestrado em Engenharia Eletrotécnica na área de especialização em Automação e Comunicações em Sistemas de Energia.

O estágio foi realizado na empresa SA – Soluções em Automação, S.A., e decorreu entre os meses de outubro de 2013 e junho de 2014, incidindo na temática do aprofundamento da formação em contexto de trabalho e desenvolvimento de soluções de automação para a indústria cerâmica, mais especificamente na área de programação de autómatos *Siemens*.

1.2. Motivação

Os principais motivos que levaram à escolha do estágio curricular foram a possibilidade de usufruir de uma formação em contexto de trabalho, a possibilidade de aplicação de conhecimentos teóricos adquiridos durante o percurso académico a um contexto mais prático e atual, a integração na dinâmica de uma empresa com largos anos de experiência na área de automação industrial, a possibilidade de interagir com novas tecnologias e "formas de fazer", bem como a perspetiva de integrar futuramente os quadros da empresa.

1.3. Objetivos

O estágio teve como objetivo o aprofundamento da formação em contexto de trabalho através da integração nas atividades da empresa, em particular nas seguintes áreas:

- Programação de autómatos;
- *Software* para automação;
- Conhecimento das soluções de autómatos *Siemens*;
- Identificação dos pontos necessários para o desenvolvimento de um programa para *Programmable Logic Controller* (PLC) dedicado à indústria cerâmica;
- Validação e comissionamento dos programas desenvolvidos;

De forma a cumprir os objetivos, foram adquiridos conhecimentos em programação de autómatos *Siemens*, variadores eletrónicos de velocidade da marca SEW *Eurodrive*, protocolos de comunicação e *hardware* de automação.

1.4. Organização do Trabalho

O estágio foi dividido em quatro fases, descritas nos capítulos 3, 4 e 5. Estruturalmente o relatório encontra-se dividido no capítulo atual e mais 5 capítulos com a estrutura que a seguir é descrita.

- O capítulo 2 aborda a revisão da literatura, onde são demonstradas outras alternativas à *Siemens* e os seus pontos a favor e contra;
- O capítulo 3 é dedicado à primeira das quatro fases do estágio, o conhecimento e aprofundamento das soluções disponibilizadas pela *Siemens*, a sua aplicabilidade, funcionalidade e mais-valias;
- O capítulo 4 corresponde à segunda e terceira fases do estágio, que se traduzem em levantamento dos pontos necessários para a correta conceção e desenvolvimento de programas para PLC de comando e controlo de máquinas para a indústria cerâmica;
- O capítulo 5 aborda a quarta fase do estágio, dar apoio ao comissionamento e validação do *software* desenvolvido, bem como apresentar os resultados obtidos e avaliação;
- No capítulo 6 encontram-se as conclusões e as sugestões para trabalho futuro.

2. COMPARAÇÃO DE CARATERÍSTICAS DOS PLC MAIS UTILIZADOS NA EMPRESA

Todas as funções foram criadas para trabalhar em autómatos da *Siemens*, este facto prende-se única e exclusivamente por esta marca ser a mais utilizada pela empresa onde se realizou o estágio. Este facto não impede que grande parte das funções sejam utilizadas em PLC de outras marcas, bastando para isso ajustar o seu código de forma a ficar compatível com autómatos de outras marcas.

Quando se fala em tecnologias de automação existe sempre a discussão do porquê de uma marca em relação a outras, ou neste caso, o porque de *Siemens* em relação a outras marcas existentes no mercado. Muitas das vezes o cliente é que escolhe a marca, não por ser a melhor ou pior, mas porque é aquela com que está mais familiarizado.

Apesar de se ter utilizado apenas e exclusivamente *Siemens* para se programar, testar e implementar as funções, durante o estágio surgiu a oportunidade de ter contacto com outras marcas, tais como *Omron* e *Schneider*. Serão estas as três marcas comparadas neste capítulo, em primeiro lugar por se ter tido contacto com elas durante o estágio e o curso, e em segundo lugar, porque a nível da indústria cerâmica estas três marcas (*Siemens*, *Omron* e *Schneider*) muito provavelmente são líderes de mercado.

A figura 2.1 mostra três autómatos existentes no mercado hoje em dia das três marcas mencionadas acima.

Figura 2.1: Exemplos de autómatos das marcas *Omron* (CP1L), *Siemens* (S7 – 1200) e *Schneider(Modicon Premium)*, da esquerda para a direita.

A comparação entre as três marcas mencionadas pode ser feia em três categorias, *hardware*, *software* e protocolos de comunicação. A questão de possuírem ou não soluções para o tipo de indústria específico não se coloca neste caso pois todas elas possuem soluções para a indústria cerâmica, que é o tipo de indústria focado neste relatório.

Começando pelo *hardware*, todas as três marcas possuem diferentes gamas de autómatos, na gama mais baixa da *Siemens* existe o LOGO que é um PLC bastante limitado no número de entradas e saídas, com velocidade de processamento lenta comparado com autómatos de gamas superiores, mas que permite ser programado sem a necessidade de recorrer a *software* específico, a programação é feita recorrendo a botões físicos existentes nele.

Na *Omron* existe o CPM2C caraterizado por ser bastante pequeno para permitir uma maior poupança de espaço, mas como os outros, bastante limitado no que toca a velocidade de

processamento e número de entradas e saídas, apesar de permitir um maior número do que o LOGO da *Siemens*. Em relação à *Schneider* temos o TWIDO, um autómato programável para controlo de máquinas simples, este autómato permite a utilização de módulos de expansão, estando limitado a um máximo de 40 *I/O*.

Na *Siemens* existe um modelo (S7-1200) uma gama acima do LOGO que também permite a utilização de módulos de expansão tal como o PLC da *Schneider* e da *Omron*. O recurso a estes modelos não é o mais indicado caso se queira controlar processos industriais pois é bastante limitado em entradas e saídas e capacidade de processamento, mas se o objetivo for o de controlo de máquinas individuais e isoladas os autómatos mencionados são sem dúvida uma opção a ter em conta.

Passando para a gama média alta, e para unidades com maior capacidade de processamento e capazes de controlar processos industriais de alguma complexidade, surge na *Scheneider* o *Modicon* M340, que é um dos modelos de gama média alta da marca e permite centenas de I/O, tempos de processamento muito mais curtos e opções de comunicação que os de gama baixa não permitem.

Na *Siemens* e com as mesmas caraterísticas do anterior temos a gama SIPLUS S7-300 mais propriamente a CPU 315-2PN/DP e na *Omron* temos entre outros o PLC CJ2M. Todas as 3 CPU apresentadas possuem caraterísticas semelhantes entre elas, podendo variar um pouco nas velocidades de processamento, no número de entradas e saídas suportadas ou nos protocolos de comunicação suportados. São estas as pequenas diferenças que juntamente com o preço levam a escolher uma marca em relação a outra.

Para finalizar temos os PLC de gama alta, estes permitem multitarefa, gestão de processos complexos e extensos, possuem grandes quantidades de memória e uma fiabilidade melhorada (utilização de redundância). Os três exemplos apresentados de seguida servem para mostrar uma opção de cada marca.

Na *Omron* no que toca a gama alta existe o CS1D, na *Siemens* temos o SIPLUS S7-400 com CPU 412-2 PN e na *Schneider* temos a gama *Modicon Quantum*.

Quando se escolhe o *hardware* para a aplicação que se deseja construir, existem muitos aspetos a ter em conta além da marca e da sua fama, tais como fiabilidade, facilidade de implementação, preço, protocolos de comunicação suportados, facilidade de expansão, etc. As três marcas apresentadas são muito semelhantes em termos de oferta de *hardware*, o que leva à consideração de outros fatores tais como a facilidade de programação, e é aí que entra o *software*.

Entenda-se que quando se refere ao *software*, estamos a falar do *software* de programação disponibilizado pelas marcas necessário na programação dos autómatos. Na *Siemens* temos o *Simatic Manager*, na *Omron* existe o *CX-One* e os seus muitos subprogramas e na *Schneider* temos o *TwidoSuite* para o PLC *Twido, Unity Pro* para a programação das CPU de gama mais alta e um sem número de *softwares* para configurar redes de comunicação, *hardware*, etc.

No aspeto de *sofware* a *Siemens* destaca-se das outras marcas devido à facilidade de configuração e à integração de todos os programas necessários numa única *suite* de

programação. Por experiência, no que toca a facilidade de implementação, a *Siemens* possui a melhor ferramenta de programação (*Simatic Manager*, ou o seu sucessor *TiaPortal*) das três marcas referidas neste capítulo.

Outro dos aspetos importantes a ter em consideração na hora de escolher a marca do PLC são os protocolos de comunicação que se deseja implementar, ou as limitações existentes no terreno que levem a escolher um protocolo em vez de outro mais rápido ou fiável.

Hoje em dia existem vários protocolos de comunicação, uns abertos e outros propriedade de determinada marca, uns mais rápidos outros mais lentos e fiáveis. Para se escolher o protocolo ajustado à aplicação que se está a desenvolver tem que se ter em conta o tipo de informação a transmitir, os destinatários, se são da mesma marca ou não, caso não sejam, se suportam os mesmos protocolos, e a velocidade necessária para garantir que não existem atrasos na execução dos processos a serem controlados.

Apenas a título de exemplo serão descritos alguns dos protocolos de comunicação permitidos por cada uma das marcas apresentadas, os protocolos apresentados nas listas seguintes não são os únicos protocolos de comunicação suportados pelas marcas mas são os mais conhecidos e utilizados.

Na *Omron* temos os seguintes protocolos de comunicação, entre outros:

- *Ethernet*;
- *DeviceNet*;
- *Profinet*;
- *ModBus*;

A *Schneider* possibilita os seguintes protocolos de comunicação, entre outros:

- *CANopen*;
- *Interbus*;
- *DeviceNet*;
- *Profibus*;

A *Siemens* permite entre outros:

- *Profibus*;
- *Profinet*;
- *Modbus*;

Por último surgem as linguagens de programação permitidas pelas marcas mencionadas, todas elas permitem praticamente todas as linguagens existentes para programação de autómatos, e a sua implementação é bastante semelhantes nas três marcas. Neste aspeto não existe grande diferença entre elas.

Tendo em conta os aspetos apresentados neste capítulo e não só, foi tomada a decisão de que as funções desenvolvidas no estágio fossem baseadas na programação de funções para autómatos da marca *Siemens*. Os motivos que levaram a escolher a *Siemens* passam por esta marca possuir uma grande fiabilidade, facilidade de programação e parametrização das redes

de comunicação e do *hardware* a instalar, grande parte dos clientes da empresa preferirem *Siemens* e porque grande parte do trabalho desenvolvido na empresa onde se realizou o estágio recorre a soluções desta marca. Assim sendo não faria muito sentido ter-se desenvolvido as funções em outra plataforma além da *Siemens*.

3. SOLUÇÃO DE *SOFTWARE* EM *SIEMENS*

Durante a fase inicial do estágio, foi adquirida formação em sistemas *Siemens* e em *software* de supervisão industrial (SCADA) *Wonderware*, este último não será abordado no presente relatório, dado que a formação adquirida teve como objetivo proporcionar uma melhor perceção da integração existente entre os autómatos e os softwares SCADA e as vantagens que a conjugação dos dois trás para os processos industriais.

A formação em *Siemens* foi dividida em três módulos, cada um destes módulos abordam partes diferentes dos sistemas *Siemens* e à medida que se foi avançando na formação, a informação exposta foi mais aprofundada e específica.

O primeiro módulo de formação percorre todo o universo *Siemens*, desde *software* a *hardware*, oferecendo uma explicação geral dos conceitos e demonstrando as diferentes soluções em termos de *hardware* oferecidas. Este módulo também fez a introdução ao *software Simatic Manager*.

O segundo módulo de formação fez uma abordagem mais detalhada da programação de autómatos *Siemens*, e deu explicações acerca de funções existentes no *Simatic Manager* e diferentes metodologias de programação e comissionamento de *hardware*.

Por último, no terceiro módulo foi abordado o conceito de funções multi-instância, endereçamento indireto (apenas possível em linguagem STL), *debugging* de erros, protocolos de comunicação e controladores PID.

Os aspetos mais importantes, e que vale a pena salientar serão abordados nos seguintes subcapítulos deste relatório.

3.1. Módulo 1

O primeiro módulo de formação da *Siemens* [1] oferece uma visão geral sobre a família de produtos e *softwares* disponibilizados pela empresa para a área de automação industrial. Na figura 3.1 é mostrado um esquema representativo da família de produtos *Siemens*.

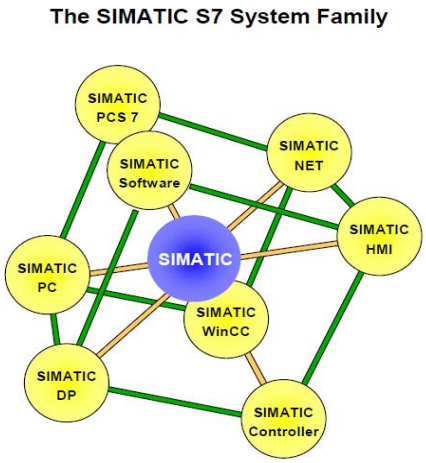


Figura 3.1: Família de soluções em automação oferecidas pela *Siemens*[1].

A família de produtos *Simatic* unifica todos os dispositivos e sistemas, tais como *hardware* e *software*, numa uniforme e poderosa plataforma.

Em relação ao *hardware*, são demonstradas as diferentes gamas de CPU, tais como o LOGO, S7-200, S7-300 (figura 3.2) e S7-400 (por ordem de capacidade de processamento). Estas CPU já sofreram atualizações para versões mais recentes, a S7-200 passou a chamar-se S7-1200 por exemplo. Dado que durante o estágio a CPU utilizada foi a S7-300, as outras unidades de processamento não serão tão referidas no decorrer deste relatório.

Figura 3.2: Imagem da CPU 315-2PN/DP da *Siemens* da gama S7-300.

A S7-300 é caraterizada por ser uma unidade compacta, oferecer uma gama bastante diversificada de modelos de processadores tais como o modelo 314, 315, 317, etc. Esta unidade possui um *slot* para cartão de memória, este cartão é onde é guardado o programa, mesmo que exista uma falha de energia o cartão guarda o *backup* e o último estado das memórias internas do tipo retentivas.

Esta CPU permite adicionar até 32 módulos de expansão sem a utilização de uma unidade de expansão (ET200S). Estes módulos podem ser de entradas e saídas digitais, entradas e saídas analógicas, módulos de comunicação, etc.

Para se poder comunicar, programar e comissionar o *hardware* referido anteriormente, é necessário uma plataforma de programação. Neste caso foi utilizado o *Simatic Manager* (figura 3.3), existindo já uma versão mais recente chamada *Tia Portal*.

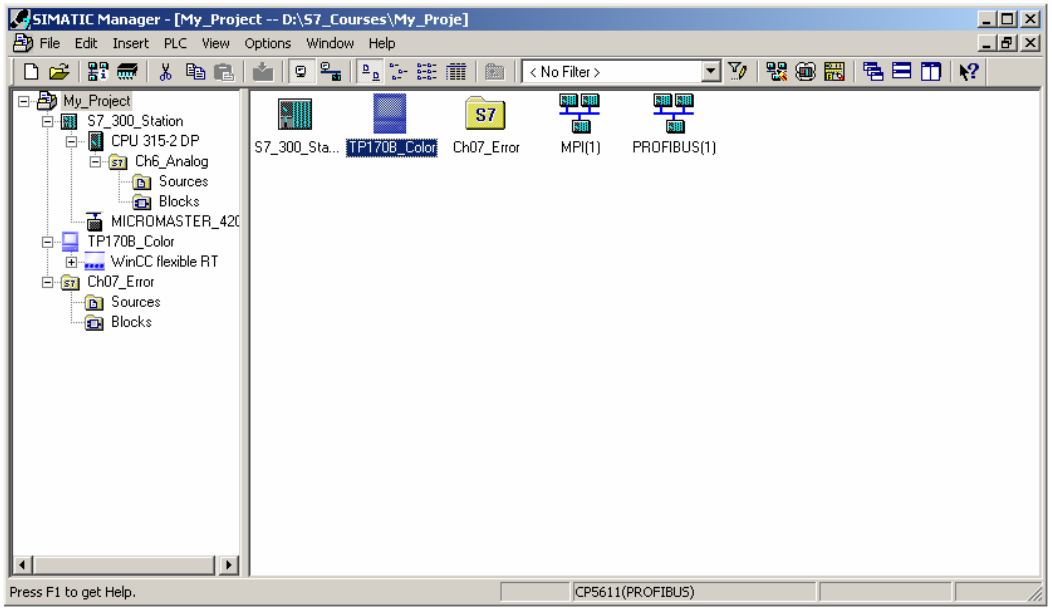


Figura 3.3: Vista geral do ambiente de trabalho do *software Simatic Manager*.

Dentro do *Simatic Manager* utilizando a ferramenta *HW Config* é possível configurar o *hardware* necessário para o projeto tal como a unidade de processamento (CPU), *I/O*, *drivers*, etc.

Durante a configuração do *hardware* é necessário ter em atenção a atribuição de endereços para os diferentes tipos de protocolos de comunicação que possam existir (*profinet*, *profibus*, etc.), o endereçamento em zona de memória das entradas e saídas dos dispositivos inseridos no projeto e a correta escolha e configuração das redes de comunicação utilizadas (figura 3.4). Esta atenção adicional serve para evitar a sobreposição de memória e tornar o acesso à mesma mais fácil.

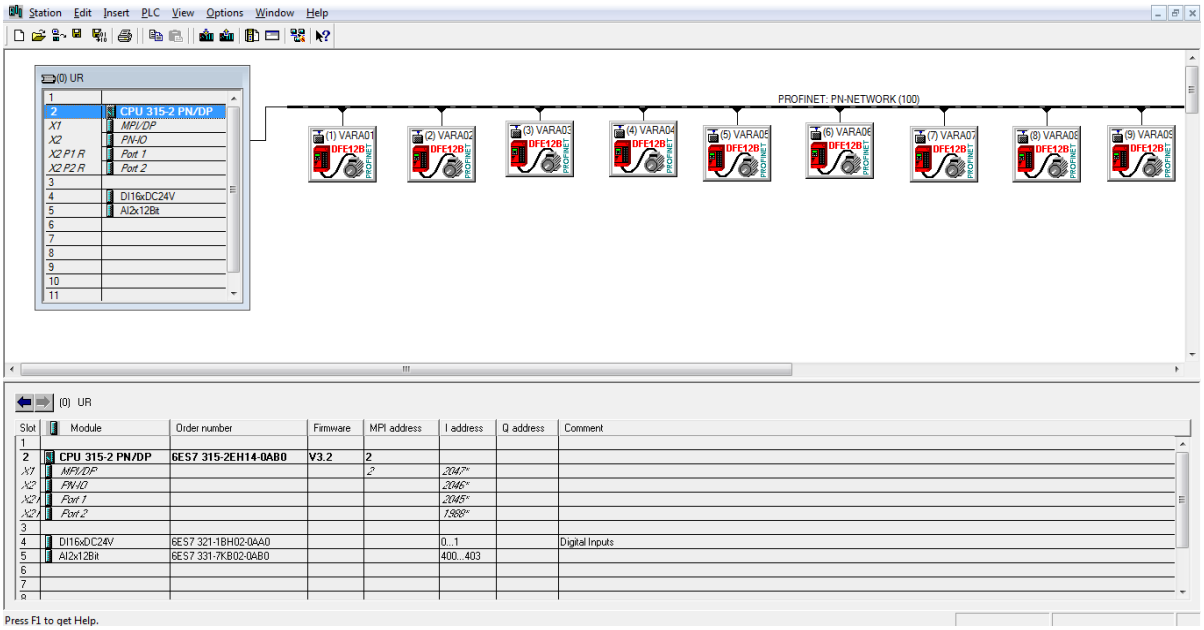


Figura 3.4: Exemplo de comissionamento de *hardware* em *Siemens*.

Após criado e configurado o *hardware* é possível aceder à estrutura de programação dentro do *Simatic Manager*. Nesta estrutura é possível encontrar uma pasta chamada *Symbol Tabel* onde se pode criar endereçamento simbólico, ou seja, a uma variável, por exemplo I0.0, pode-se atribuir um nome e começar a chamar a entrada I0.0 por esse nome. Também é possível encontrar uma pasta chamada *Source Files* (figura 3.5), onde é possível importar e guardar funções de outros projetos, tornando possível utilizar essas mesmas funções no projeto atual.

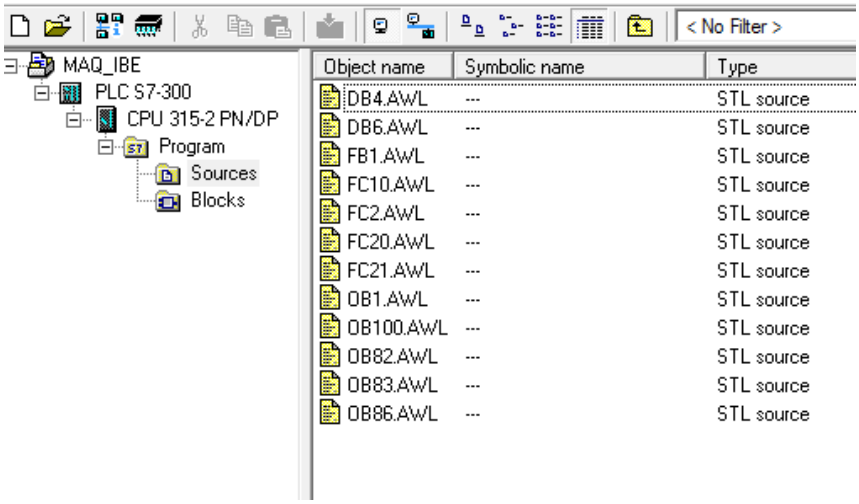


Figura 3.5: Aspeto dos ficheiros do tipo *Source File* importados pelo *Simatic Manager*.

Por último encontra-se a pasta com o nome de *Blocks*. Dentro desta pasta está o programa que irá controlar o sistema industrial. Dentro deste programa estão os OB, FC, FB, DB, etc. utilizados. Na figura 3.6 é possível ver parte de um programa criado para PLC *Siemens*.

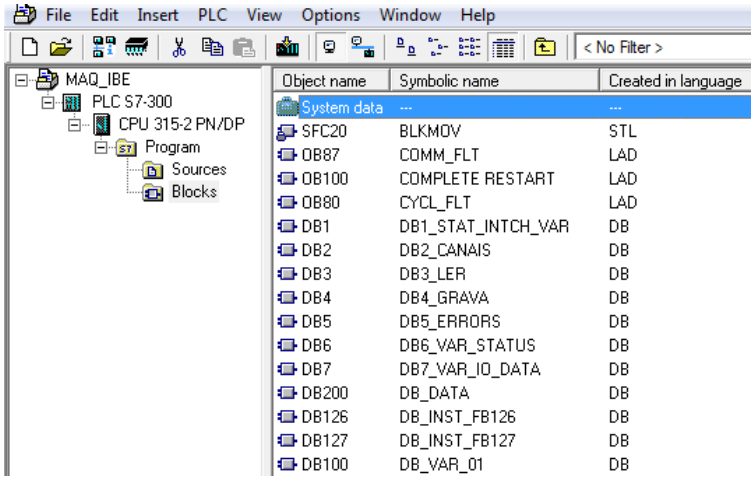


Figura 3.6: Exemplo do interior da pasta *Blocks* do *Simatic Manager*.

- OB – *Organisation Block*

Os blocos de organização (OB) formam o *interface* entre o sistema operativo da CPU e o programa do utilizador. O OB1 é o único OB que é chamada ciclicamente pelo sistema sem que tenha que se configurar qualquer parâmetro.

- FC – *Function*

As FC contêm funcionalidades parciais dos programas. Como é possível adicionar parâmetros, estas tornam-se ideais para executar partes do programa que sejam repetidas muitas vezes, por exemplo fazer um *scaling* ao valor lido por um sensor de temperatura.

- FB – *Function Block*

Os FB oferecem as mesmas condições que as FC, sendo a principal diferença entre ambos a possibilidade de os FB terem a sua própria área de memória criada na forma de DB.

- DB – *Data Block*

Data Blocks são áreas de memória no programa criado pelo utilizador de forma estruturada, e às quais é possível aceder diretamente.

- SFB – *System Function Block*

Os SFB possuem as mesmas características dos FB, sendo a única diferença entre ambos o facto de os FB serem criadas pelo programador, e os SFB estarem inseridos nas bibliotecas de funções existentes no *software*.

- SFC – *System Function*

As SFC possuem as mesmas características das FC, sendo a única diferença entre ambas o facto de as FC serem criadas pelo programador, e as SFC estarem inseridas nas bibliotecas de funções existentes no *software*.

Em relação às linguagens de programação referidas durante a formação, e apesar de os *softwares* da *Siemens* possibilitarem a utilização de outras linguagens, as linguagens de programação mais utilizadas no decorrer do estágio foram *Ladder* e STL. O *Ladder* (figura 3.7) é uma linguagem de programação de autómatos baseada em objetos e universalmente utilizada, desde a *Siemens*, passando pela *Omron* e continuando para todas ou quase todas as marcas de autómatos.

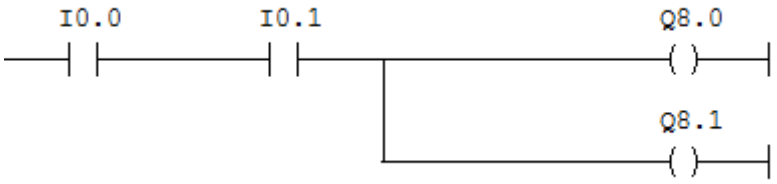


Figura 3.7: Exemplo de programa em linguagem *Ladder*.

O STL (figura 3.8) é um tipo de linguagem caraterizada por ser escrita em linha de comandos sequencial. Ao ser composta por uma sequência de comandos escritos, torna possível criar certos comandos de código que utilizando a linguagem *Ladder* não seria possível. As suas principais desvantagens prendem-se com o facto de ser de difícil compreensão para alguém que não trabalhe todos os dias com ela, o diagnóstico dos programas em tempo real e observação do estado atual das variáveis não é tão amigável quanto a linguagem *Ladder*.

```
A     I     0.0
A     I     0.1
=     Q     8.0
=     Q     8.1
```

Figura 3.8: Exemplo de programa em linguagem STL.

Quando se fala de redes de comunicação em *Siemens* a mais conhecida e utilizada é sem dúvida a rede *profibus* [2]. A rede *profibus* é constituída por uma unidade definida como *master*, normalmente o autómato, e por um conjunto de unidades definidas como *slaves*, módulos de expansão, *drivers* de motores, etc. Todas os aparelhos ligados à rede *profibus* possuem um endereço único, que é definido no comissionamento do *hardware*.

A figura 3.9 mostra uma ficha *profibus*.

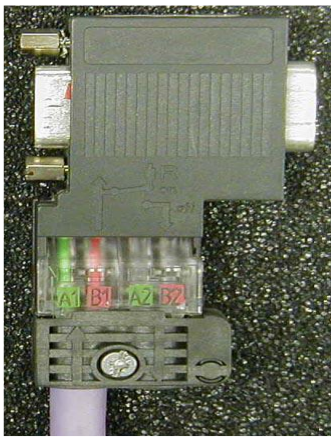


Figura 3.9: Ficha de comunicação *profibus*.

Durante a formação, o protocolo de comunicação mais explorado foi o *profibus*. A escolha de não abordar tão profundamente os outros protocolos de comunicação foi feita com base na premissa de que no decorrer do estágio o *profibus* seria o protocolo de comunicação utilizado nas aplicações desenvolvidas.

Os equipamentos HMI, apesar de existirem e de serem referidos durante a formação, não serão abordados no decorrer deste relatório, dado que não foram desenvolvidas aplicações para este tipo de equipamento. De referir apenas que é necessário um *software* específico chamado *WinCC* para programar estes equipamentos, caso o projeto seja feito em *Simatic Manager*. Se for utilizado o *Tia Portal* para desenvolver o projeto o *software WinCC* deixa de ser necessário.

3.2. Módulo 2

Durante o segundo módulo de formação [3] aprofundaram-se os temas introduzidos no módulo anterior. A começar pelo hardware e de como é feita a leitura de valores analógicos obtidos em cartas de medição ou a interligação entre as *control words* dos *drivers* de controlo e o programa do autómato. Para que tal seja possível, quando se faz o comissionamento do *hardware* existem variáveis que indicam as zonas de memória onde é feita a interligação com os módulos de

expansão e de que tipo de interligação se trata, entrada de informação ou saída de informação. Caso se trate de valores de entrada, no programa temos que ler variáveis do tipo PIW, se forem variáveis de saída, trata-se de variáveis do tipo PQW (figura 3.10). Por exemplo PIW256 e PQW 256.

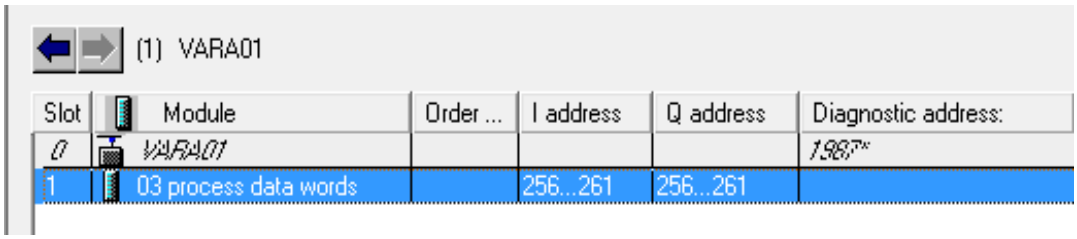


Figura 3.10: Exemplo de endereçamento do *hardware* em autómatos *Siemens*.

A forma como a CPU executa o programa é de extrema importância pois indica a sequência de leitura e execução das funções, e qual a prioridades entre elas. Durante o primeiro ciclo de funcionamento é executado um OB específico (este OB pode variar entre modelos de PLC), caso seja do interesse do programador executar algum código apenas no primeiro ciclo do autómato, esse código deve ser escrito dentro desse OB, ou em funções depois chamadas nesse bloco de organização.

Existe um OB (OB1) que é chamado em todos os ciclos do autómato. Dentro deste OB deve ser colocado o corpo principal do programa, aquela parte que é desejável que esteja sempre a executar e a testar as suas condições.

Os blocos de organização de interrupção, interrompem o funcionamento cíclico do programa e executam o código inserido dentro deles, (OB 35), código este que terá prioridade em relação ao que se encontra dentro do OB1. Os tempos de interrupção podem ser definidos nas propriedades da CPU, e indicam de quanto em quanto tempo é que o OB de interrupção é chamado.

Em relação aos FB, FC, etc, estes apenas serão executados se forem chamados dentro de algum dos OB referidos anteriormente, quer seja o OB1, os OB de interrupção ou o de primeiro ciclo. A figura 3.11 mostra um esquema com o ciclo de funcionamento de uma CPU da *Siemens*.

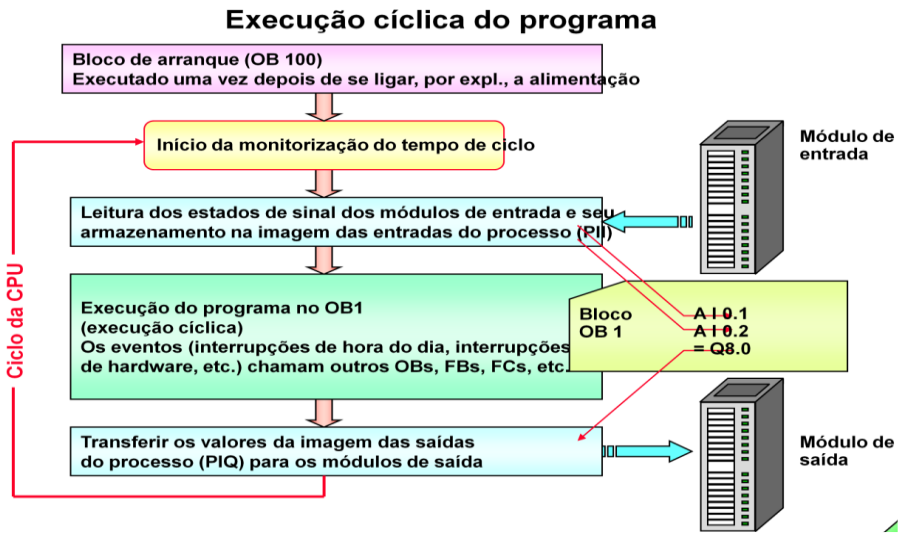


Figura 3.11: Esquema demonstrativo da execução de um ciclo da CPU de um autómato *Siemens*[3].

Nem todas as funções têm necessariamente que ser chamadas dentro de um OB, elas podem também ser executadas dentro de outras funções, tornando-se assim subfunções. Estas subfunções normalmente são funções que não necessitam de ser executadas ciclicamente, estando a sua execução pendente do estado de alguma variável ou conjunto de variáveis.

Uma grande vantagem que esta estrutura de funções permite, é a execução repetida da mesma função para controlar os mesmos processos em equipamentos diferentes sem que para isso seja necessário criar uma função para cada equipamento. Por exemplo, uma função que controle a velocidade de um motor pode ser chamada 10 vezes e controlar a velocidade de 10 motores diferentes. Estas funções são chamadas de multi-instância (figura 3.12).

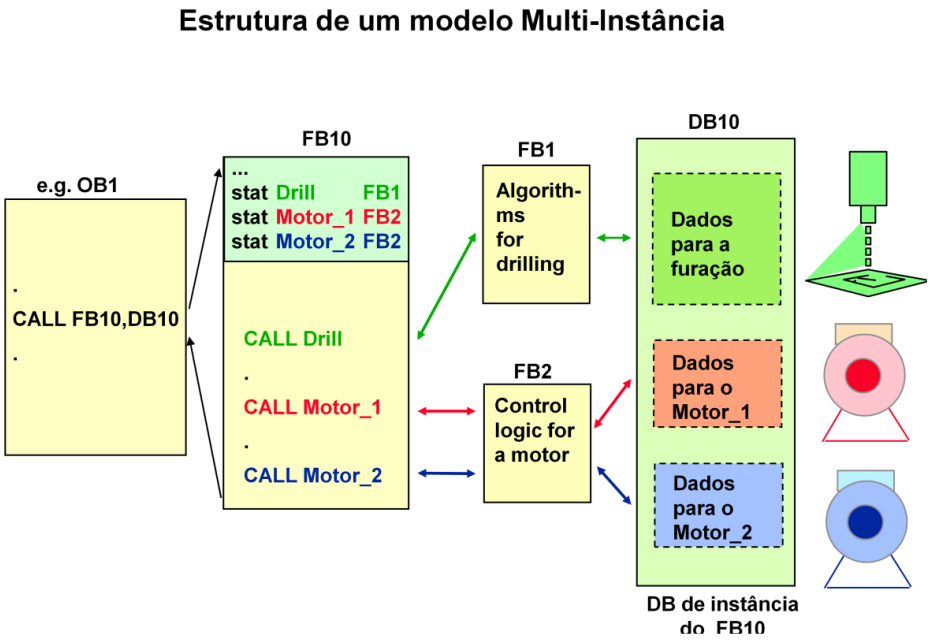


Figura 3.12: Estrutura de um FB multi-instância [3].

Para se poder criar um programa de automação funcional, o endereçamento é de extrema importância. Em *Siemens* existem três tipos de endereçamento, o endereçamento direto no qual se indica diretamente a zona de memória a ler ou escrever (mw100, m10.0, etc.), o endereçamento simbólico no qual se atribui a uma zona de memória um nome ou mnemónica e depois se passa a aceder a essa zona utilizando o nome definido, sendo esta atribuição de nomes a zonas de memória feita na *symbol table*, por último a terceira forma de se usar endereçamento em *Siemens* é por endereçamento indireto, que através de um ponteiro podemos aceder a zonas de memória sem ter que as chamar diretamente. O endereçamento indireto em *Siemens* não pode ser feito recorrendo à linguagem *Ladder*, tem que ser feito em linguagem textual como STL.

Outros temas foram abordados no módulo dois de formação, mas a sua utilidade durante o decorrer do estágio não se revelou importante a ponto de serem inseridos no relatório final.

3.3. Módulo 3

O módulo de formação três [4] começa com o tema da possibilidade de se ter funções em múltipla instância dentro de um programa de autómato. Desta forma é possível não subcarregar a CPU com a execução cíclica de funções que apenas são chamadas algumas vezes no decorrer do normal funcionamento da máquina a programar. Com o multi-instânciamento também é possível uma melhor organização do código e das variáveis de memória utilizadas por este, pois estas podem ser todas guardadas dentro do mesmo *Data Block* (DB) e acedidas para diagnóstico nesse mesmo DB.

Quando se fala de variáveis dos FB e FC fala-se de variáveis definidas pelo programador aquando da criação da função e que apenas dizem respeito a ela, tais como variáveis de entrada, saída, estáticas e temporárias (figura 3.13). As variáveis temporárias perdem o seu valor a cada ciclo do autómato, ao contrário das estáticas que o conservam até que este seja alterado na execução do código.

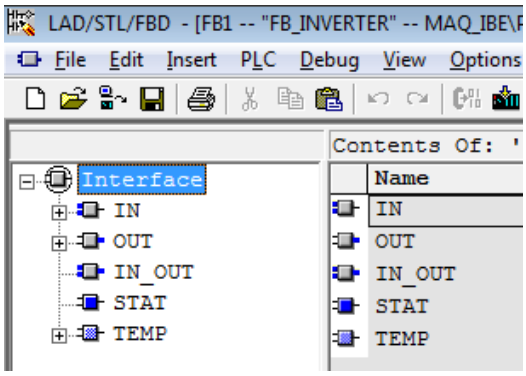


Figura 3.13: Tipos de variáveis possíveis de encontrar no interior de uma função.

Não é possível criar variáveis estáticas nas FC, pois este tipo de função não possui instânciamento, apenas são permitidas variáveis de entrada, saída e temporárias. Já nos FB também é possível definir variáveis estáticas uma vez que estas funções possuem DB de

instância. Quando se fala de instânciamento fala-se de atribuir um DB ao FB quando este é chamado dentro de um bloco de organização, ou de outra função do programa.

Também é possível chamar um FB dentro de outro FB, sendo que nesta situação não é requerido que se atribua um DB ao FB chamado dentro do outro, ficando as memórias declaradas dentro do segundo FB instanciadas no DB do primeiro (figura 3.14).

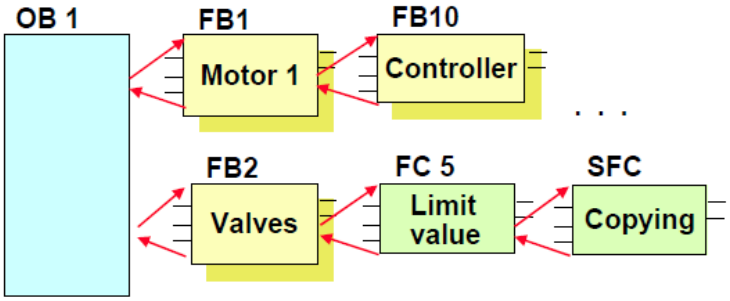


Figura 3.14: Ordem de execução de diferentes funções ao longo de um ciclo de execução da CPU [4].

Continuando com o tema da declaração de variáveis, é importante determinar o tipo de variáveis que são necessárias durante a construção das funções, o tipo de variáveis que é permitido inserir e a quantidade de memória que estas ocupam. Na lista seguinte estão indicados os principais tipos de variáveis e a memória requerida por cada um deles.

- *Bool – 1 bit;*
- *Byte – 8 bits;*
- *Word – 16 bits;*
- *DWord – 32 bits;*
- *Char – 1 Byte;*
- *Int – 1 Word;*
- *Dint – 1 DWord;*
- *Real – 1 DWord*;
- *Array* [*bool, byte, int*, …] – tipo de memória utilizado no *array* vezes a quantidade de vezes que essa memória existe, *Array* [1..10] *of Int*, ocupa 1*10 *words*;
- *Struct* – o tamanho depende dos elementos definidos dentro da estrutura;
- Etc.

Cada tipo de variável possui uma quantidade definida de memória que pode utilizar, sendo assim não é possível declarar uma variável como sendo *Dint* e copiar o seu conteúdo para outra do tipo *Int* e esperar que o valor seja copiado sem perda de informação. É de extrema importância estar atento à conjugação de tipos de variáveis e ao tipo de parâmetros que elas suportam que lhes sejam introduzidos.

Definidas as variáveis e o seu tipo, muitas vezes torna-se necessária a interação entre funções recorrendo a estas mesmas variáveis. Para que tal seja possível estas tais variáveis podem ser definidas como sendo de entrada, saída ou de entrada e saída, onde a interligação é feita na área onde se chama a função. Na figura 3.15 é demonstrada a interligação entre variáveis de entrada e de saída do FB1 e memórias de entrada e saída do PLC.

```
CALL  FB     1 , DB2
 Start   :=I0.0
 Stop    :=I0.1
 Motor_on:=Q12.0
 Speed   :=QW14
```

Figura 3.15: Chamada do bloco função FB1 em linguagem STL.

Outra forma de se interligarem variáveis de funções diferentes é recorrendo ao multi-instanciamento, onde uma função é chamada dentro de outra e as suas variáveis passam a estar acessíveis no DB de instância da primeira função. De salientar que as variáveis temporárias não podem ser acedidas fora da própria função em que são declaradas.

Recorrendo ao endereçamento indireto também é possível a passagem de parâmetros entre funções, bastando saber onde se encontra a variável a ler ou escrever e o *offset* desta em relação ao início da zona de memória em que se encontra. A título de exemplo, e pressupondo que se quer ler o valor inteiro que se encontra dentro do DB1, DB de instância do FB1, e que este se encontra com um *offset* de 100. Dentro do FB2 basta para tal abrir o DB1 e criar um ponteiro que aponte para a zona 100 do DB e de seguida ler a variável para uma variável pertencente ao FB2. Mostra-se na figura 3.16 um excerto de código STL onde está demonstrado um ponteiro para escrever o valor 0 numa variável existente no DB de outro FB.

```
      L     #AUX_NUM_OF_VAG
      L     2
      *I
      L     118
      +I
      SLD   3
      L     P#0.0
      +D
      T     #PONT1
      L     0
      T     DBW [#PONT1]
      R     #BIT_AUX_FIM_LINHA
      CLR

_03:  CLR
      NOP   0
```

Figura 3.16: Exemplo de um ponteiro em linguagem STL.

Para finalizar o capítulo de formação, é necessário falar de uma das funções de sistema possíveis de encontrar nas bibliotecas do *software* de programação, que foi utilizada e testada exaustivamente no decorrer do estágio, que é a função PID (FB41).

Para o controlo de processos, o PID é das melhores ferramentas que se pode utilizar, e recorrendo ao PID da *Siemens* apenas é necessário atribuir os parâmetros necessários para que o controlo feito pelo PID seja adequado ao processo que se quer controlar e que esteja dentro das margens definidas para o mesmo.

A figura 3.17 mostra a chamada do PID dentro de uma FB.

Existem algumas considerações a ter em conta para que o PID funcione normalmente, sendo a mais importante o facto de este ter que ser chamado dentro de um OB de interrupção, OB35 por exemplo, com ciclo de interrupção fixo. Este ciclo de interrupção tem que ser igual ao tempo inserido no parâmetro de entrada "*Cycle*" do PID.

Depois de terminado o módulo de formação três e de serem feitos alguns exercícios práticos acerca dos conceitos explicados, deu-se por terminada a formação em *Siemens* e passou-se para a fase seguinte do estágio

```
CALL   "CONT_C" , DB230
 COM_RST :=
 MAN_ON  :=
 PVPER_ON:=
 P_SEL   :=
 I_SEL   :=
 INT_HOLD:=
 I_ITL_ON:=
 D_SEL   :=
 CYCLE   :=T#100MS
 SP_INT  :=
 PV_IN   :=
 PV_PER  :=
 MAN     :=
 GAIN    :=
 TI      :=
 TD      :=
 TM_LAG  :=T#2S
 DEADB_W :=
 LMN_HLM :=
 LMN_LLM :=
 PV_FAC  :=
 PV_OFF  :=
 LMN_FAC :=
 LMN_OFF :=
 I_ITLVAL:=
 DISV    :=
 LMN     :=
 LMN_PER :=
 QLMN_HLM:=
 QLMN_LLM:=
 LMN_P   :=
 LMN_I   :=
 LMN_D   :=
 PV      :=
 ER      :=
NOP   0
```

Figura 3.17: Chamada da função de PID existente na biblioteca do *Simatic Manager*.

4. FUNÇÕES DESENVOLVIDAS

4.1. Introdução – Indústria Cerâmica

Antes de se começar a falar dos programas desenvolvidos propriamente ditos, é necessário apresentar alguma contextualização acerca do tipo de indústria que é a cerâmica, e das máquinas utilizadas por esta indústria e que foram de alguma forma abordadas durante o estágio.

A indústria cerâmica, mais propriamente a cerâmica de fabrico de tijolos (figura 4.1) para a construção civil, apesar de em Portugal ter sofrido bastante com a estagnação da construção e com a crise económica dos últimos anos, em países em desenvolvimento apresenta uma grande oportunidade de negócio, uma vez que muitos destes países se encontram a ter um grande *boom* na construção.

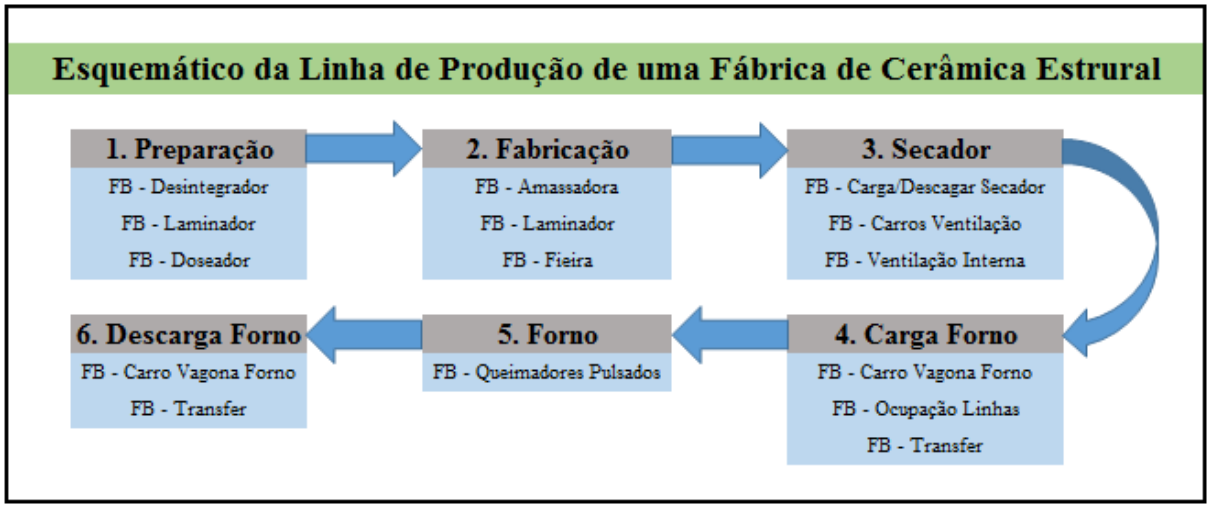


Figura 4.1: Esquemático da linha de produção de uma cerâmica de tijolo com indicação dos vários FB criados para cada zona.

O processo de fabrico do tijolo é caraterizado por ser um processo de fabrico contínuo e que não pode ter muitas interrupções pelo meio, caso isso aconteça corre-se o risco de parar a fábrica completamente durante bastante tempo. Para que tal não aconteça é necessário contar com máquinas fiáveis e com processos de automação igualmente fiáveis e redundantes a falhas. Quando se fala em fiabilidade no mundo da automação muitas vezes surge o nome *Siemens* associado.

O processo de fabrico do tijolo começa, como muitos outros, pela matéria-prima, neste caso argila, a argila até estar pronta para formar o tijolo propriamente dito precisa passar por um processo de transformação, onde é amassada, triturada, laminada e finalmente extrudida na forma do tijolo por uma máquina chamada de fieira.

Após a argila ganhar a forma do tijolo segue para um secador, onde irá reduzir a sua humidade para valores aceitáveis para poder entrar no forno. Na saída do secador o tijolo é transportado para um conjunto de pinças que o coloca nas vagonas do forno, vagonas estas que irão encaminhar o tijolo para o forno.

Depois de cozido o tijolo é empilhado em paletes, embalado e expedido para os clientes. As funções que irão ser apresentadas seguirão a ordem das máquina durante o fabrico do tijolo, para tentar demonstrar a envolvência do estágio e a sua contribuição para a colocação em operação de uma fábrica do género descrito anteriormente.

Não será feita uma explicação muito detalhada do interior das funções porque em primeiro lugar iria estender em demasia o relatório, e em segundo porque o código criado e a forma de funcionamento de algumas máquinas é propriedade das empresas, o que impossibilita a sua divulgação neste relatório.

4.2. Telas de Transporte

A matéria-prima (argila) durante o processo de preparação é transportada de máquina em máquina recorrendo a telas transportadoras (figura 4.2).

Figura 4.2: Tela de transporte de matéria-prima.

Considerando que a zona de preparação da argila pode chegar a ter dezenas de telas de transporte e que algumas delas podem se deslocar em ambos os sentidos, foi criada uma função (FB) padrão que desse para controlar quase todas as telas de transporte existentes na zona de preparação da argila. Apenas o controlo de um tipo de tela não é abrangido pela função, uma vez que possui movimento de translação com controlo de posição o que iria tornar a função muito complexa quando fosse usada para controlar telas normais.

Para colocar a tela em funcionamento é utilizado um motor de indução trifásico, e é precisamente o controlo do funcionamento deste motor que a função faz.

A função criada permite movimentar a tela em ambos os sentidos, ligar os espalhadores, caso existam, na extremidade da tela e gerar alarmes sempre que alguma proteção falhe ou o funcionamento não esteja de acordo com o esperado. Na figura 4.3 é mostrado um pequeno excerto do código da função de controlo das telas.

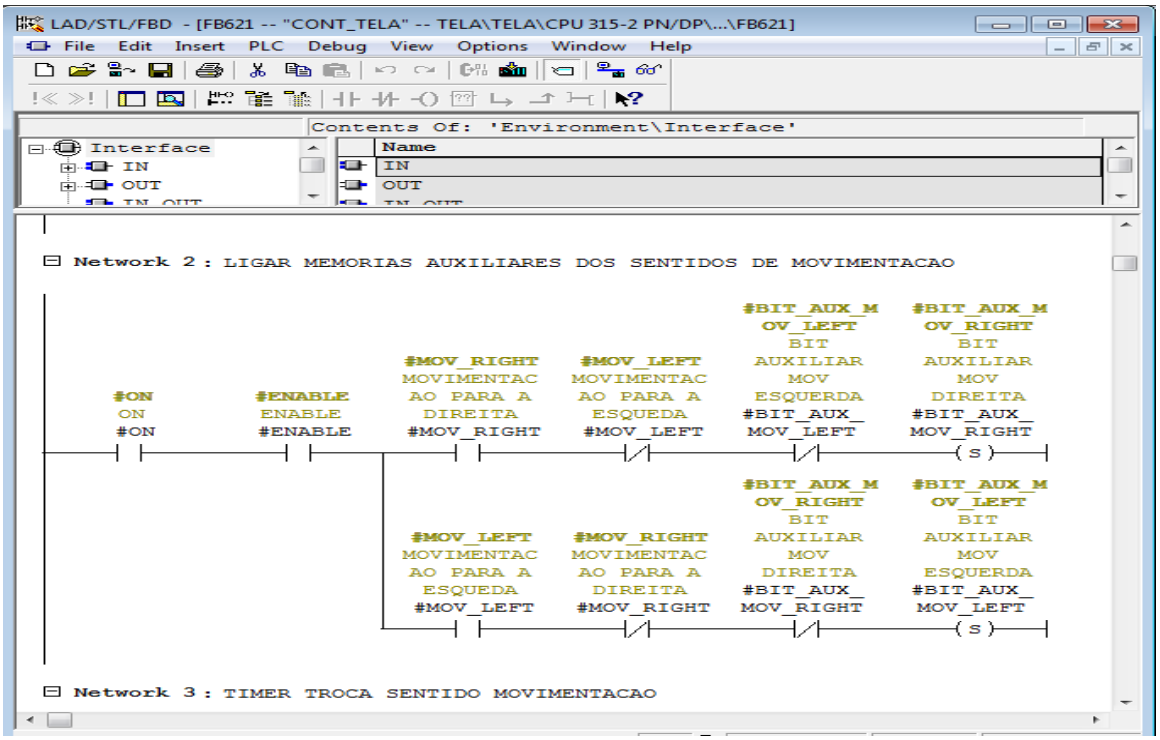


Figura 4.3: Excerto da função de controlo das telas de transporte.

De salientar que as ordens de arranque e de paragem das telas, encontram-se fora da FB, como esta pode ser utilizada para controlar mais do que uma Tela, e como as ordens de arranque e paragem não são iguais para todas, essa parte do programa é exterior à FB.

O *grafcet* de nível 1 desta função encontra-se anexo a este relatório (Anexo A), o *grafcet* demonstra o funcionamento do movimento da tela de transporte.

Após concluída, a função ficou com as seguintes entradas e saídas:

- **Entradas**
  - *Enable* – Condição on/off da função; (*Bool*)
  - *Mov_Right* – Ordem de movimentação da tela para a direita; (*Bool*)
  - *Mov_Left* – Ordem de movimentação da tela para a esquerda; (*Bool*)
  - *Spreader 1* – Ordem para ligar o espalhador 1; (*Bool*)
  - *Spreader 2* – Ordem para ligar o espalhador 2; (*Bool*)

  Se a tela não possuir espalhadores estas duas entradas devem ser ignoradas.

  - *Motor_Started* – Indicação de que o motor arrancou, esta condição apenas se usa quando utilizado arrancador suave. Se algum tempo após a ordem de arranque do motor este não arrancar, são desligadas as saídas e gerado alarme de erro; (*Bool*)
  - *Termic_Protection* – Proteção térmica do motor e do freio caso exista; (*Bool*)
  - *With_Soft_Starter* – Indicação de que a tela possui arrancador suave; (*Bool*)
  - *Reset_Failure* – *Reset* da saída de falha da FB; (*Bool*)
- **Saídas**
  - *R_Mov_Right* – Ligar relé de movimentação para a direita do motor; (*Bool*)

- *R_Mov_Left* – Ligar relé de movimentação para a esquerda do motor; (*Bool*)
- *R_Spreader 1* – Ligar espalhador 1; (*Bool*)
- *R_Spreader 2* – Ligar espalhador 2; (*Bool*)

Se a tela não possuir espalhadores, estas duas saídas devem ser ignoradas.

- *Failure* – Indicação de falha na tela; (*Bool*)

Com esta FB é possível fazer a interligação entre as condições de funcionamento e as saídas de controlo dos motores para vários tipos de tela.

4.3. Controlo do Desintegrador

O desintegrador é uma máquina existente na preparação da argila que permite desintegrar e homogeneizar a argila. Esta FB permite controlar desintegradores das marcas *Bedeschi* (figura 4.4) e *Verdés*.

Figura 4.4: Imagem de um desintegrador da marca *Bedeschi*.

O desintegrador, como mostra a figura, possui dois cilindros que rodam em sentidos opostos. Estes cilindros são acionados por dois motores independentes que podem ter um simples sistema de arranque em estrela-triângulo ou ser acionados por variadores eletrónicos de velocidade.

Além de controlar o funcionamento da máquina e de determinar se a máquina deve parar devido a ocorrência de determinados erros, a função também possibilita que seja feita a retificação do cilindro maciço em condições de segurança, pois bloqueia o arranque enquanto as condições de retificação estejam ativas.

Devido ao facto de a função necessitar de ter muitos parâmetros de entrada foi utilizado um byte para cada cilindro, onde cada bit corresponde a determinado parâmetro predefinido, dentro da função esses bits são observados e consoante o seu estado é feita a ação programada. Esta solução foi adotada de forma a tornar a função mais compacta no que toca ao número de parâmetros de entrada.

Na figura 4.5 é mostrado um pequeno excerto do código da função de controlo do desintegrador.

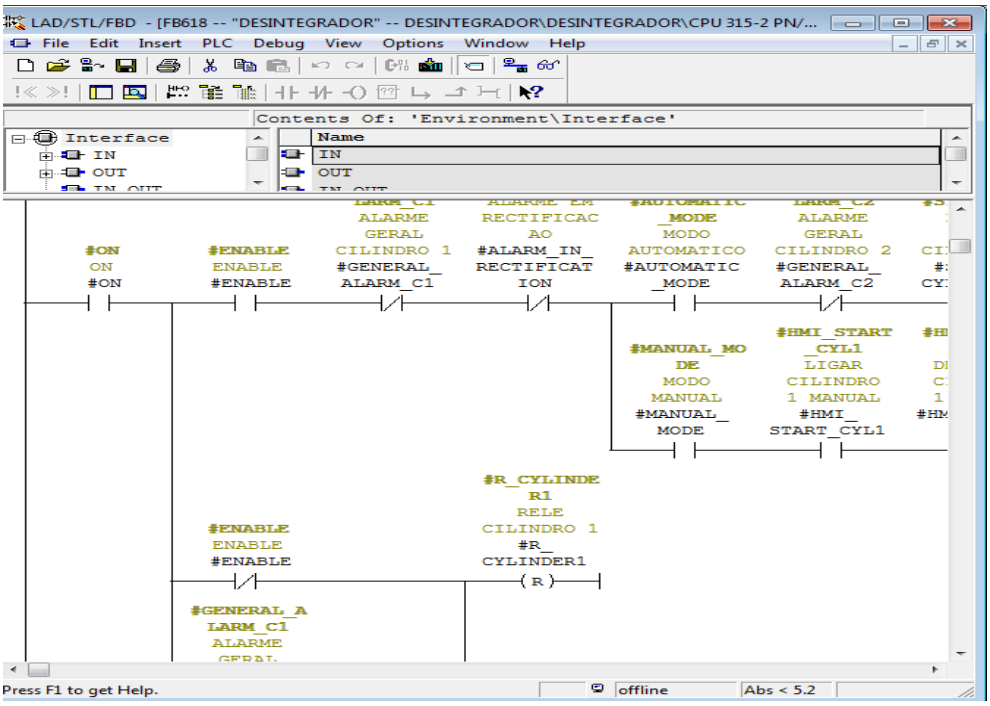


Figura 4.5: Excerto da função de controlo do desintegrador.

Os alarmes gerados pela FB não se encontram definidos como parâmetros de saída desta, estão antes inseridos na memória estática da função, e podem ser acedidos recorrendo ao DB de instância da FB. Assim como os limites de consumo definidos para os motores, têm que ser escritos diretamente dentro da DB de instância. Optou-se por colocar estes parâmetros apenas como variáveis do tipo estático, porque 90% das vezes apenas são lidos/escritos recorrendo a consolas de interface HMI e para a programação das consolas torna-se mais fácil ler e escrever diretamente dos DB.

O desintegrador também possui um sistema de raspadores no cilindro maciço para retirar alguma argila que esteja colada na superfície deste, estes raspadores também são controlados pela função.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Após concluída a FB, esta ficou com a seguinte estrutura de entradas e saídas, estas entradas e saídas são o que permite uma interligação mais direta e óbvia com o resto do programa.

- **Entradas**
  - *Enable* – Condição on/off da função; (*Bool*)
  - *Automatic Mode* – Quando colocado a 1 coloca a função em modo automático, e a 0 indica que o controlo é manual; (*Bool*)
  - *Start_Cylinder1* – Ordem para arrancar com o cilindro 1; (*Bool*)
  - *Start_Cylinder2* – Ordem para arrancar com o cilindro 2; (*Bool*)
  - *Start_Scrapers* – Ordem para ativar os raspadores; (*Bool*)
  - *Parameters_C1* – Byte contendo os parâmetros do cilindro 1; (*Byte*)

- *Parameters_C2* – Byte contendo os parâmetros do cilindro 2; (*Byte*)
- *Start_Rectification* – Ordem para que se inicie a retificação do cilindro; (*Bool*)
- *Cylinder1_Consumption* – Consumo do cilindro 1, este valor é comparado com o consumo máximo e caso esteja acima mais do que alguns segundos é parada a máquina e gerado um alarme; (*Real*)
- *Cylinder2_Consumption* – Consumo do cilindro 2, este valor é comparado com o consumo máximo e caso esteja acima mais do que alguns segundos é parada a máquina e gerado um alarme; (*Real*)
- *Pressure_Switch_Air* – Pressóstato de ar dos raspadores, bit normalmente fechado, caso vá a um mais do que alguns segundos ativa um alarme; (*Bool*)
- *Vev_Ok* – Indicação de que o variador, caso exista, se encontra sem erros; (*Bool*)
- *Reset_Alarms* – *Reset* dos alarmes da FB; (*Bool*)

- ***Byte* de parâmetros dos Cilindros 1 e 2**
  - Bit 0 – *Cylinder Started* – Indicação de que o cilindro arrancou;
  - Bit 1 – *Diferencial Protection* – Estado da proteção diferencial do motor do cilindro;
  - Bit 2 – Temperatura Ok – Indicação de que o sensor de temperatura do cilindro está ok;
  - Bit 3 – Térmico Cilindro – Proteção térmica do motor do cilindro encontra-se ok;
  - Bit4 – Detetor de Rotação – Indicação de que o cilindro se encontra em movimento, se houver ordem de funcionamento e não for detetada rotação é gerado alarme;
  - Bit5 – Com VEV – Indicação de que o cilindro é acionado por variador eletrónico de velocidade;
- **Saídas**
  - *R_Cylinder1* – Ligar cilindro 1; (*Bool*)
  - *R_Cylinder2* – Ligar cilindro 2: (*Bool*)
  - *R_cylinder2_Vel2* – Ligar cilindro 2 na velocidade de retificação; (*Bool*)
  - *R_Scrapers* – Ligar raspadores; (*Bool*)
  - *General_Alarme_C1* – Indicação de que o cilindro 1 possui algum alarme; (*Bool*)
  - *General_Alarme_C2* – Indicação de que o cilindro 2 possui algum alarme; (*Bool*)

A função acima descrita permite o controlo do mesmo tipo de máquina mas de dois fabricantes diferentes, e torna mais fácil a implementação e colocação em funcionamento da mesma, inserida num projeto de controlo de toda a parte da preparação do material da fábrica.

4.4. Controlo do Laminador

O laminador (figura 4.6) é em muito parecido com o desintegrador, tanto a nível de funcionamento com a nível estético. A grande diferença entre ambos reside no facto de o laminador possuir ambos os cilindros maciços, enquanto o desintegrador possui um cilindro maciço e superfície lisa outro de superfície não lisa com saliências metálicas.

Figura 4.6: Imagem de um laminador.

Tal como a função descrita no subcapítulo 4.3, também esta função permite controlar máquinas da marca *Verdés* e da marca *Bedeshi*.

Mesmo o funcionamento do laminador sendo muito parecido com o do desintegrador, possui algumas condições diferentes.

O FB do laminador apesar de não ser muito mais complexo do que o do desintegrador, possui mais parâmetros e condições capazes de gerar alarmes e obrigar à paragem da máquina, tornando a função um pouco mais extensa em número de *networks*. Na figura 4.7 é possível observar parte de uma destas *networks*.

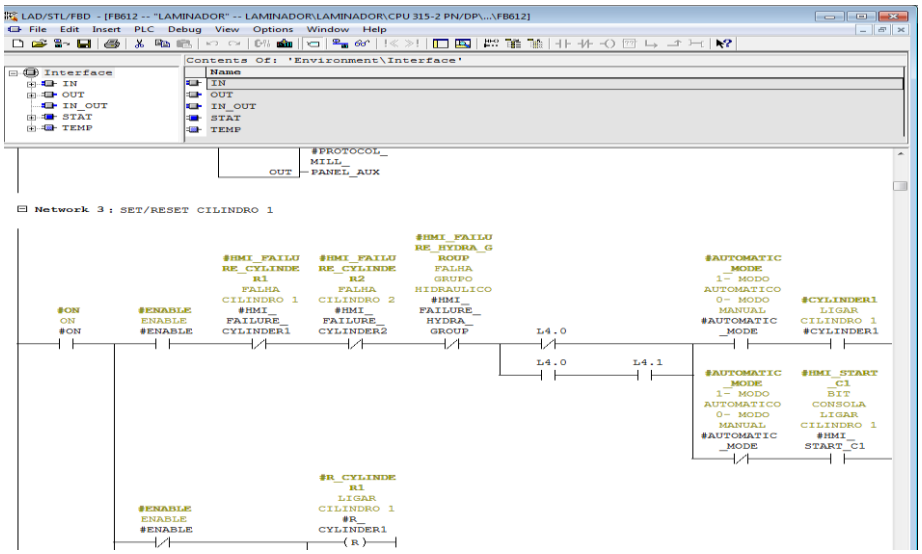


Figura 4.7: Excerto da função de controlo do laminador.

Como é necessário ajustar e manter constante a distância entre os cilindros, o laminador possui uma centralina hidráulica que mantém os cilindros posicionados corretamente. O funcionamento e deteção de problemas da centralina também é controlado pelo FB de controlo do laminador.

Com o desgaste dos cilindros torna-se necessário fazer a retificação dos mesmos. Para este processo o laminador possui um componente chamado de galga, que é responsável por fazer o cilindro andar a baixas rotações enquanto é retificado. Na entrada da função existe um byte de parâmetros para a galga.

Uma vez que o FB tem a capacidade de controlar máquinas de marcas diferentes, existe um byte na entrada deste que diz respeito aos bits de protocolo recebidos do quadro elétrico próprio do laminador, estes protocolos, caso existam, têm que ser levados em conta na hora de colocar a máquina em funcionamento.

Como no desintegrador, também na função do laminador, os diferentes alarmes encontram-se dentro do DB de instância do FB, e podem ser acedidos dentro desta.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Os parâmetros de entrada e saída criados para a FB são os seguintes,

- **Entradas**
  - *Enable* – Condição on/off do FB de controlo do laminador; (*Bool*)
  - *Automatic_Mode* – Identifica o modo de funcionamento em que se encontra a máquina, a 1 em automático e a 0 em manual; (*Bool*)
  - *Cylinder 1* – Condição para o funcionamento do cilindro 1; (*Bool*)
  - *Cylinder 2* – Condição para o funcionamento do cilindro 2; (*Bool*)
  - *Scrapers* – Condição para funcionamento dos raspadores; (*Bool*)
  - *Parameters_C1* – Parâmetros de funcionamento do cilindro 1; (*Byte*)
  - *Parameters_C2* – Parâmetros de funcionamento do cilindro 2; (*Byte*)
  - *Parameters_Galga* – Parâmetros de funcionamento da Galga; (*Byte*)
  - *Parameters_Protocol_Panel_Lami* – Parâmetros do protocolo entre o quadro do laminador e o PLC que controla o funcionamento do mesmo; (*Byte*)
  - *Limit_Switch_Scrapers* – Detetor de raspadores gastos; (*Bool*)
  - *Cylinder1_Consumption* – Consumo atual do cilindro 1; (*Real*)
  - *Cylinder2_Consumption* – Consumo atual do cilindro 2; (*Real*)
  - *Pressure_Switch_Air* – Estado do pressóstato de ar da máquina; (*Bool*)
  - *Pressure_Switch_Centr* – Estado do pressostadto da centralina; (*Bool*)
  - *Thermal_Protection_Centr* – Proteção térmica do motor da centralina; (*Bool*)
  - *Det_Min_Level_Oil* – Detetor de nível de óleo mínimo; (*Bool*)
  - *Reset_Alarms* – Bit de *reset* dos alarmes da FB de controlo do laminador; (*Bool*)

Os *bytes* de parâmetros de ambos os cilindros, da centralina e do protocolo do quadro do laminador são descritos de seguida. De salientar que nem todos os laminadores possuem protocolo entre o seu quadro elétrico e o autómato que o controla e que nem todos possuem galga para a retificação.

- ***Byte* de parâmetros dos cilindros 1 e 2**
  - Bit 0 – *Cylinder Started* – Indicação de que o cilindro arrancou;
  - Bit1 – *Det Cylinder Recessed* – Detetor de cilindro recuado, para que seja possível retificar o cilindro, este tem que se encontrar recuado;
  - Bit 2 – *Diferencial Protection* – Estado da proteção diferencial do motor do cilindro;
  - Bit 3 – Temperatura *Ok* – Indicação de que o sensor de temperatura do cilindro está ok;
  - Bit 4 – Térmico Cilindro – Proteção térmica do motor do cilindro encontra-se *ok*;

- ***Byte* de parâmetros da galga**
  - Bit 0 – *With Galga* – Indicação de que o laminador possui galga;
  - Bit1 – *Start Galga* – Ordem para colocar a galga em funcionamento;
  - Bit 2 – *Advance Galga* – Avançar a galga, até esta encostar na polia do cilindro;
  - Bit 3 – *Retreat Galga* – Recuar a galga até a posição de repouso;
  - Bit 4 – *Security Ok* – Detetor de segurança da galga;
  - Bit 5 – *Pressure_Switch_Centr_Low* – Indicação de que a pressão na centralina da galga se encontra baixa.

- ***Byte* de parâmetros do protocolo do quadro do laminador**
  - Bit 0 – *With Protocol From Electrical Panel Off The Mill* – Indicação de que o laminador possui quadro próprio com protocolo de comunicação para o PLC.
  - Bit1 – *P_In_Electrical Panel in Automatic* – O bit quando a 1 indica que o quadro está ligado em automático, e quando a 0 em manual;
  - Bit 2 – *P_In_Electrical Panel Ok* – Indicação de que o quadro elétrico não se encontra em erro;

Por último temos as saídas do *Function Block.*

- **Saídas**
  - *R_Cylinder1* – Ligar cilindro 1; (*Bool*)
  - *R_Cylinder2* – Ligar cilindro 2: (*Bool*)
  - *R_Scrapers* – Ligar raspadores; (*Bool*)
  - *R_Centr* – Ligar a centralina; (*Bool*)
  - *Valve_Cylinder* – Abrir válvula de ar do laminador; (*Bool*)
  - *Galga_Active* – Indicação de que a galga se encontra ligada; (*Bool*)
  - *Advance_Galga* – Ordem de avanço da galga; (*Bool*)
  - *Retreat_Galga* – Ordem de recuo da galga; (*Bool*)
  - *Can_Rectify* – Indicação de que se pode iniciar a retificação dos cilindros; (*Bool*)
  - *General_Alarme_C1* – Indicação de que o cilindro 1 possui algum alarme; (*Bool*)
  - *General_Alarme_C2* – Indicação de que o cilindro 2 possui algum alarme; (*Bool*)

4.5. Controlo da Amassadora

A função de controlo da amassadora (figura 4.8), exatamente como as duas funções referidas anteriormente, permite controlar máquinas das marcas *Verdés* e *Bedeschi*. Este FB é bastante menos extensa e de funcionamento mais simples do que as anteriores, já que a máquina em si também é de funcionamento mais simples.

Figura 4.8: Imagem de uma amassadora.

A amassadora possui um motor com embraiagem, que aciona dois eixos helicoidais sem fim, que rodam em sentidos opostos e lado a lado, amassam a argila e empurram-na de uma extremidade para a outra da amassadora, onde sai e continua com o processo de preparação do material. Também possui um sistema que introduz água na mistura existente dentro da amassadora.

Mesmo que o motor esteja ligado, a embraiagem da amassadora só atraca quando todas as condições de funcionamento desta estão respeitadas e existe material dentro da mesma.

Na figura 4.9 é mostrado um pequeno excerto do código da função de controlo da amassadora.

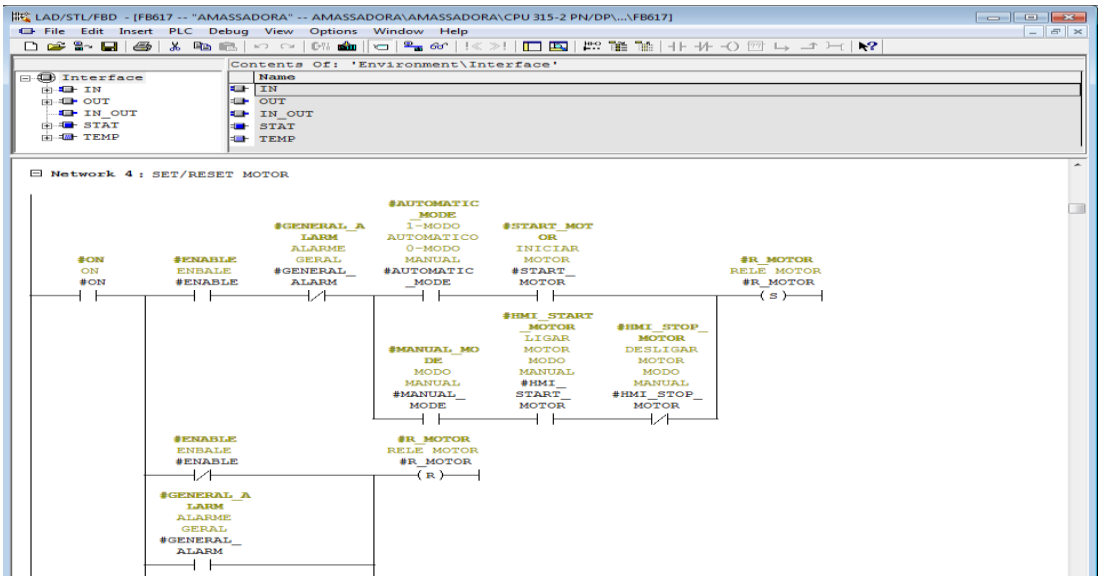


Figura 4.9: Excerto da função de controlo da amassadora.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

O FB referido neste capítulo, e à semelhança de outras, também possui um byte como parâmetro de entrada, este byte possui os parâmetros correspondentes a proteções da amassadora.

- **Entradas**
  - *Enable* – Condição on/off do FB de controlo da amassadora; (*Bool*)
  - *Automatic_Mode* – Identifica o modo de funcionamento da máquina, a 1 em automático e a 0 em manual; (*Bool*)
  - *Start_Motor* – Condição para o funcionamento do motor; (*Bool*)
  - *Engaging_the_Clutch* – Condição para embraiar a amassadora; (*Bool*)
  - *Water_Valve* – Ordem de abertura da válvula de água; (*Bool*)
  - *Det_Material* – Detetor de material na amassadora, indica se a máquina está a receber material, caso não esteja, esta é parada; (*Bool*)
  - *Parameters_Mixer* – Parâmetros de funcionamento da amassadora; (*Byte*)
  - *Motor_Consumption* – Consumo atual do motor, caso passe do limite estipulado durante mais que alguns segundos é desligada a máquina; (*Real*)
  - *Reset_Alarms* – *Reset* dos alarmes do FB de controlo da amassadora; (*Bool*)

- ***Byte* de parâmetros da Amassadora**
  - Bit 0 – *Motor Started* – Indicação de que o motor arrancou;
  - Bit1 – *Motor Diferential Protection* – Proteção diferencial da amassadora;
  - Bit 2 – *Motor Temperature Ok* – Indicação de que a temperatura se encontra ok;
  - Bit 3 – *Motor Termic Protection* – Proteção térmica do motor;
  - Bit 4 – *Pressure Switch Air* – Estado do pressóstato de ar da amassadora;
  - Bit 5 – Fluxostato – O estado deste sensor apenas é considerado quando a lubrificação estiver ativa;
  - Bit 6 – Termostato *Oil* – Estado do termostato do óleo, a 0 termostato ok, a 1 termostato em alarme;

Em relação à ordem de lubrificação da máquina, esta encontra-se ligada enquanto o motor estiver em funcionamento e não existir nenhum alarme na função.

- **Saídas**
  - *R_Motor* – Ligar motor da amassadora; (*Bool*)
  - *R_Clutch* – Ordem para embraiar; (*Bool*)
  - *R_Lubrification* – Ligar lubrificação; (*Bool*)
  - *V_Water* – Abrir a válvula da água; (*Bool*)
  - *General_Alarm* – Indicação de que o FB se encontra com alarme; (*Bool*)

4.6. Controlo do Doseador

O doseador (figura 4.10) é um género de depósito temporário da matéria-prima e que permite colocar nas telas o material de forma mais doseada e controlada. A função explicada neste capítulo permite controlar doseadores da *Bedeschi* e da MetalCértima.

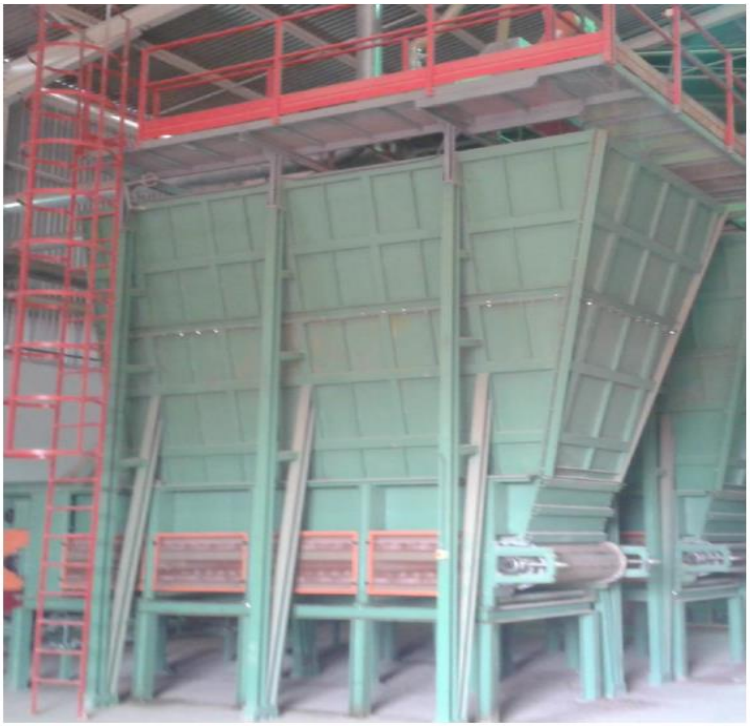


Figura 4.10: Imagem de um doseador MetalCertima.

Estes doseadores possuem dois detetores de material incorporados, um de ultrassom e outro do tipo interruptor de nível máximo. Estes detetores servem para determinar se o doseador está no nível mínimo ou máximo, vazio ou cheio. Caso o doseador esteja cheio dá sinal para que a matéria-prima comece a ser colocado noutro doseador, e se estiver vazio avisa que pode receber matéria--prima.

No fundo do doseador existe uma tela responsável por encaminhar o material até à extremidade do doseador, na qual existe um conjunto de gadanhos que auxiliam no despejo do material para a tela seguinte. A movimentação da tela é feita com o auxílio de correntes e de rodas dentadas, e o seu controlo é feito dentro da FB de controlo do doseador. Na figura 4.11 é mostrado um pequeno excerto do código da função de controlo do doseador.

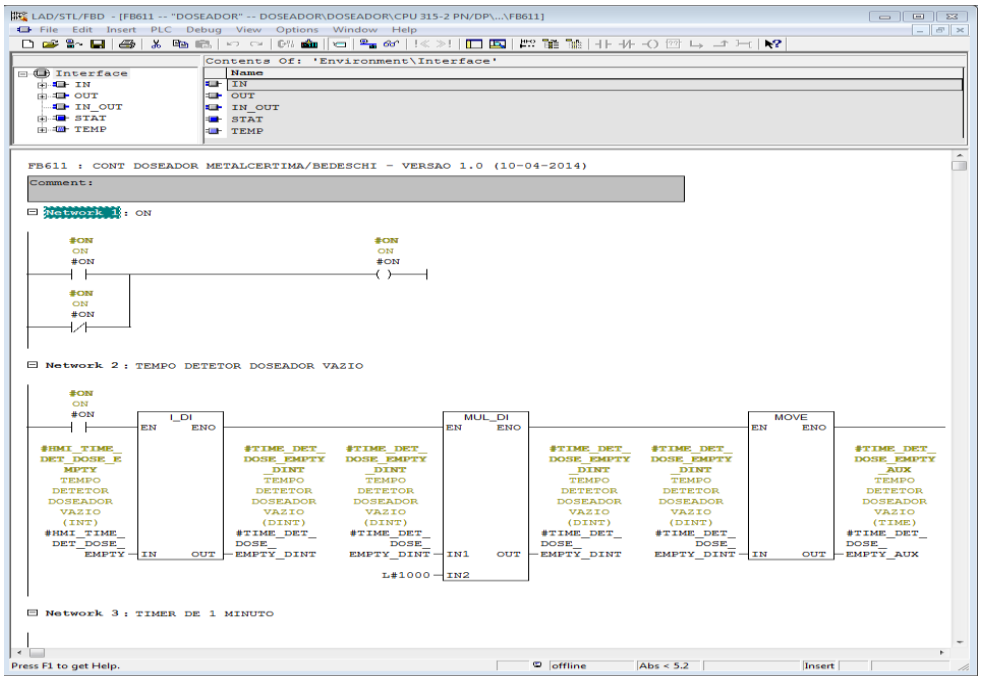


Figura 4.11: Excerto da função de controlo do doseador.

Quando o doseador se encontra com a indicação de vazio, mas é do interesse do operador arrancar com ele nesse estado, basta colocar a entrada de automático a 1. Mas se passados dois minutos (este tempo é configurável pelo operador) e o doseador continuar vazio este é parado e volta a ficar com a saída que indica que se encontra vazio ligada.

A FB também controla a lubrificação das correntes da tela do doseador, caso exista, esta lubrificação só é feita com a tela em movimento e de x em x tempo.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Após finalizada a função, esta ficou com os seguintes parâmetros de entrada e de saída.

- **Entradas**
  - *Enable* – Condição on/off do FB de controlo do doseador; (*Bool*)
  - *Automatic_Mode* – Identifica o modo de funcionamento da máquina, a 1 em automático e a 0 em manual; *(Bool*)
  - Doseador – Condição para o funcionamento do doseador; (*Bool*)
  - *Diggers* – Condição para o funcionamento dos gadanhos; (*Bool*)
  - *With_Lubrification* – Indicação de que o doseador possui lubrificação nas correntes da tela; (*Bool*)
  - *Ignore_Det_Ultrasound* – Caso a máquina não possua detetor de material por ultrassom, colocando a entrada a 1, o estado das entradas do detetor é ignorado; (*Bool*)
  - *Det_Rot_Tela* – Indicação de que a tela se está a movimentar, se existir ordem de marcha para a tela e esta não se movimentar, é gerado alarme; (*Bool*)
  - *Det_Ultrasound_A* – Primeiro dos dois sinais enviados pelo detetor de ultrassom; (*Bool*)

- *Det_Ultrasoun_B* – Segundos dos dois sinais enviados pelo detetor de ultrassom; (*Bool*)

A conjugação do estado dos dois sinais anteriores permite saber o estado do doseador no que diz respeito ao nível de material no seu interior;

- *Det_level_High_High* – sinal do detetor de nível máximo de material do doseador; (*Bool*)
- *Det_Without_Mat* – Detetor existente na saída do doseador que determina se o doseador tem ou não material; (*Bool*)
- *Det_Low_Level_Oil* – Indicação de que o óleo se encontra com um nível muito baixo; (Bool)
- *Det_Rot_Diggers* – Indicação de que os gadanhos se encontram em funcionamento; (*Bool*)
- *Speed_Auto* – Velocidade a enviar para o VEV de controlo da tela quando o doseador se encontra em automático; (*Real*)
- *Speed_Man* – Velocidade a enviar para o VEV de controlo da tela quando o doseador se encontra em manual; (Real)
- *Vev_Ok* – Indicação de que o variador não possui erros; (*Bool*)
- *Termic_Prot_Doseador* – Proteção térmica do doseador; (*Bool*)
- *Reset_Alarms* – Bit de *reset* dos alarmes da função; (*Bool*)

- **Saídas**
  - R_Doseador – Ligar doseador; (*Bool*)
  - *R_Forced_Vent* – Ligar a ventilação forçada do motor do doseador; (*Bool*)
  - *R_Diggers* – Ligar movimentação dos gadanhos; (*Bool*)
  - *R_Cent_Lub* – Ligar motor da centralina de lubrificação; (*Bool*)
  - *Open_Valv_Lub* – Ordem de abertura da válvula de lubrificação; (*Bool*)
  - *Doseador_Full* – Indicação de que o doseador está cheio; (*Bool*)
  - *Doseador_Minimum* – Indicação de que o doseador se encontra no nível mínimo; (*Boo*l)
  - *Doseador_Empty* – Indicação de que o doseador se encontra vazio; (*Bool*)
  - Speed – Velocidade desejada para a tela do doseador; (Real)
  - *General_Alarm* – Indicação de que existe alarmes na função de controlo do doseador; (*Bool*)

Grande parte das saídas necessita de uma conjugação específica de fatores para que sejam ativadas. Estes fatores dependem de temporizadores, contadores internos ou de variáveis de entrada da função.

4.7. Controlo da Fieira

Foram criadas dois FB para controlo de fieiras, um para as fieiras da marca *Verdés* (figura 4.12) e outro para a marca *Bedeschi*. Como as duas funções são bastante extensas e parecidas em termos de funcionamento e de entradas e saídas, apenas será explicado o funcionamento da função que controla as fieiras da *Verdés*. Optou-se por separar o controlo das fieiras de ambas as marcas em funções diferentes devido ao facto de terem muitos parâmetros e muitas máquinas associadas a cada uma o que levaria a estender em demasia a função.

Figura 4.12: Imagem de uma fieira da marca *Verdés*.

A fieira é o equipamento responsável por transformar a argila em tijolos recorrendo a um processo de extrusão.

Antes de começar a falar da função propriamente dita, optou-se por fazer um pequeno apanhado de todos os equipamentos que estão associados à fieira e que são também eles controlados pela função. Devido ao facto de a fieira ser um equipamento bastante grande e com uma inércia elevada é usada uma embraiagem. A sequência de acoplamento da embraiagem depende do tipo de acionamento que o motor tenha, com ou sem variador eletrónico de velocidade. Com variador, a fieira embraia com o motor parado, sem variador a fieira embraia com o motor à velocidade nominal.

Existe uma bomba de vácuo que é responsável por manter o vácuo dentro da camara da fieira, o arranque e paragem desta bomba está interligado com o estado de funcionamento da fieira.

A redutora que interliga a fieira ao motor, possui lubrificação própria e constante. Esta lubrificação também é controlada dentro do FB, assim como a lubrificação a *diesel* da fieira. A lubrificação a *diesel* possui uma centralina própria.

Como pode ser necessário adicionar água à matéria-prima na altura em que esta entra na fieira, o controlo da válvula de água também é feito no interior da função. Na figura 4.13 é mostrado um pequeno excerto do código da função de controlo da fieira.

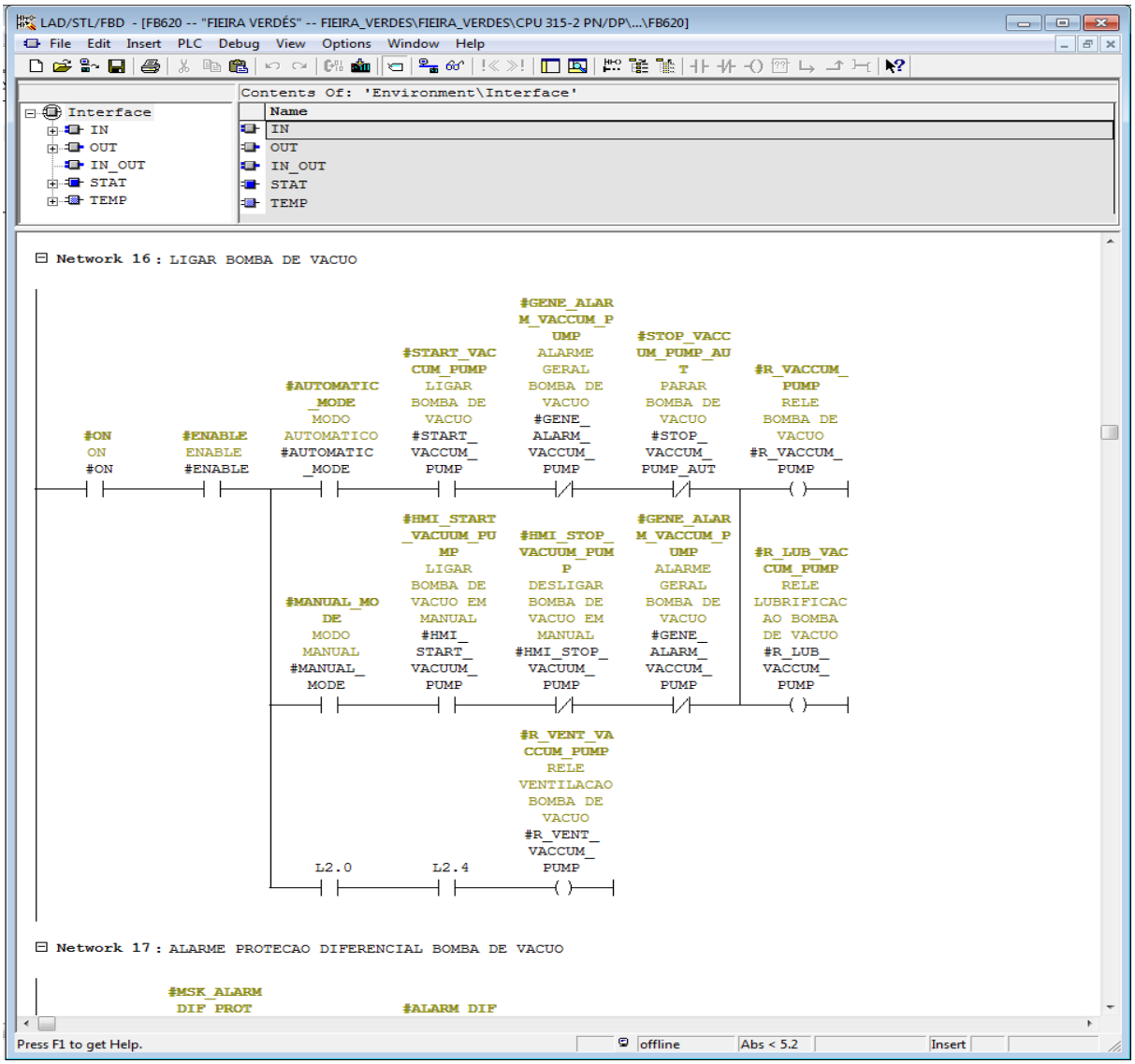


Figura 4.13: Excerto da função de controlo da fieira.

Durante o arranque da fieira, esta precisa esperar que a bomba de vácuo esteja a trabalhar para poder arrancar, e se a bomba de vácuo parar de repente, passado algum tempo a fieira também para. Caso a fieira esteja parada há mais do que o tempo definido pelo operador, a bomba de vácuo também se desliga.

Existem dois veios com sem-fins dentro da fieira, estes veios servem para empurrar o material para o bocal da máquina onde se encontra o molde. Estes veios possuem detetores de rotação que mandam parar a máquina se os veios não rodarem quando deviam.

O circuito hidráulico da fieira é controlado por uma centralina hidráulica, o funcionamento desta também é controlado pelo FB de controlo da fieira.

Os tempos de funcionamento e os alarmes dos equipamentos estão colocados dentro do DB de instância do FB. É possível aceder diretamente a estes valores através do DB, sem haver necessidade de abrir a função.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Para se tentar minimizar o comprimento do FB no ambiente de programação, alguns dos equipamentos possuem os seus parâmetros inseridos dentro de um byte de parâmetros, como já foi feito em funções anteriores.

- **Entradas**
  - *Enable* – Condição on/off do FB de controlo da fieira; (*Bool*)
  - *Automatic_Mode* – Identifica o modo de funcionamento da máquina, a 1 em automático e a 0 em manual; (*Bool*)
  - *Start_Motor* – Condição para o funcionamento do motor da fieira; (*Bool*)
  - *Engaging_the_Clutch* – Condição para embraiar a fieira; (*Bool*)
  - *Parameters_Fieira – Byte* com os parâmetros da fieira; (*Byte*)
  - *Start_Vaccum_Pump* – Condição de funcionamento da bomba de vácuo; (*Bool*)
  - *Parameters_Vaccum_Pump* – Byte com os parâmetros da bomba de vácuo; (*Byte*)
  - *Det_Material* – Estado do detetor de material na fieira; (*Bool*)
  - *Open_Water* – Condição para a abertura da válvula de água; (*Bool*)
  - *Diesel_Lub* – Quando a 1 liga a lubrificação *diesel*, a 0 desliga; (*Bool*)
  - *Pres_Switch_Red_Lub* – Pressóstato da lubrificação da redutora; (*Bool*)
  - *Pres_Switch_Air* – Pressóstato de ar da fieira; (*Bool*)
  - *Centr_Security* – Condição para ativar e desativar a centralina de segurança; (*Bool*)
  - *Pres_Switch_Cojin* – Pressóstato do cojin (sistema de veios existentes dentro da fieira); (*Bool*)
  - *Pres_Switch_Cojin_Seg* – Pressóstato de segurança do cojin; (*Bool*)
  - *Without_Vev* – Controlo do acionamento do motor feito sem recurso a variador; (*Bool*)
  - *Motor_Consumption_Fi* – Consumo do motor da fieira; (*Real*)
  - *Motor_Consumption_Vac_P* – Consumo do motor da bomba de vácuo; (*Real*)
  - *Reset_Alarms* – Bit de *reset* dos alarmes da FB; (*Bool*)

- ***Byte* de parâmetros da fieira**
  - Bit 0 – *Motor Started* – Indicação de que o motor arrancou;
  - Bit 1 – *Diferential Protection* – Proteção diferencial da fieira;
  - Bit 2 – *Temperature Ok* – Indicação de que a temperatura se encontra ok;
  - Bit 3 – *Termic Protection* – Proteção térmica do motor;
  - Bit 4 – Protocolo Fieira pode Embraiar – Indicação de que estão reunidas as condições para que a fieira possa embraiar;
  - Bit 5 – Det Rotação Veio 1 – Detetor de rotação do veio 1 da fieira;
  - Bit 6 – Det Rotação Veio 2 – Detetor de rotação do veio 2 da fieira;
  - Bit 7 – Det Porta Aberta – indica que a porta da camara da fieira se encontra aberta;

- ***Byte* de parâmetros da bomba de vácuo**
  - Bit 0 – *Pump Started* – Indicação de que a bomba arrancou;
  - Bit 1 – *Diferential Protection* – Proteção diferencial da bomba de vácuo;
  - Bit 2 – *Temperature Motor Ok* – Indicação de que a temperatura do motor se encontra ok;
  - Bit 3 – *Termic Protection* – Proteção térmica do motor da bomba de vácuo;
  - Bit 4 – *Start Ventilation* – Indicação de que a ventilação da bomba de vácuo se encontra ligada;
  - Bit 5 – *Temperature Vac_Pump* – Indicação de que a temperatura da bomba de vácuo se encontra acima do limite;

A centralina de segurança é ligada sempre que o pressóstato do cojin vai a 0. Esta é uma medida de segurança para o bom funcionamento da fieira.

- **Saídas**
  - *R_Motor_Fi* – Ligar motor da fieira; (*Bool*)
  - *R_Clutch* – Ligar embraiagem; (*Bool*)
  - *R_Open_Water* – Abrir válvula de água; (*Bool*)
  - *R_Lub_Red* – Ligar a lubrificação da redutora; (*Bool*)
  - *R_Cent_Lub_Diesel* – Ligar centralina da lubrificação *diesel*; (*Bool*)
  - *R_Cent_Security* – Ligar centralina de segurança da fieira; (*Bool*)
  - *R_Vaccum_Pump* – Ligar bomba de vácuo; (*Bool*)
  - *R_Lub_Vaccum_Pump* – Ligar lubrificação da bomba de vácuo; (*Bool*)
  - *R_Vent_Vaccum_Pump* – Ligar ventilação da bomba de vácuo; (*Bool*)
  - *Gene_Alarm_Fieira* – Indicação de que a fieira possui alarmes; (*Bool*)
  - *Gene_Alarm_Vaccum_Pump* – Indicação de que a bomba de vácuo possui alarmes; (*Bool*)

4.8. Controlo da Carga e Descarga das Vagonas do Secador

Após a fieira, o tijolo segue para o secador, onde vai perder a maior parte da humidade que possui. A forma de o material viajar pelo secador é em vagonas específicas para esse efeito. O tipo de secador referido neste subcapítulo é o secador semi-contínuo. Neste tipo de secador a saída fica do lado oposto da entrada e o material viaja pelo secador de forma lenta, podendo demorar mais do que 24 horas a atravessá-lo.

As máquinas que fazem a carga e a descarga (figura 4.14) do material das vagonas são iguais, apenas muda a sequência de movimentos e a sua ordem.

De forma a melhorar e facilitar a programação da máquina que faz a carga ou descarga do material das vagonas do secador, foi criada uma função única para esse efeito. Além de a função servir tanto para a carga como para a descarga, também não está limitada ao número de prateleiras da vagona, que pode variar de fábrica para fábrica, nem à distância entre prateleiras.

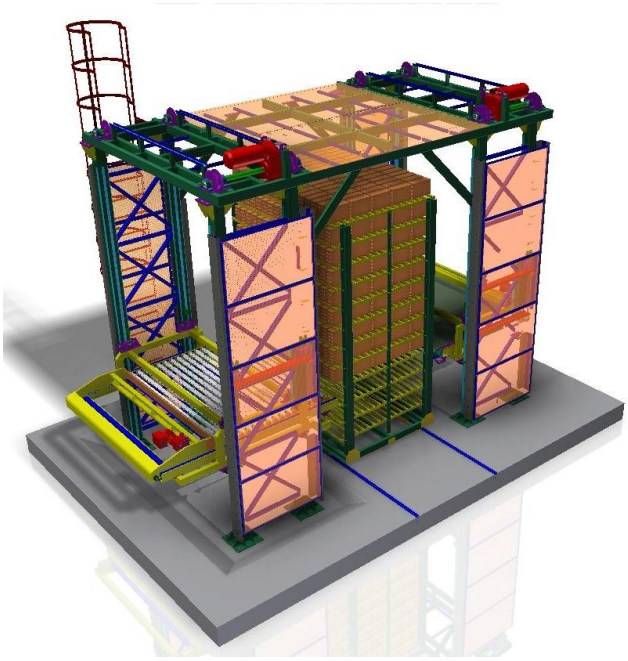


Figura 4.14: Ilustração de uma máquina de carga e descarga das vagonas do secador.

A máquina que carrega e descarrega as vagonas do secador é caraterizada por ter duas componentes móveis, uma de cada lado da vagona. A primeira leva ou trás o material até à posição da prateleira de destino, enquanto a segunda, que possui uns garfos que entram no interior da vagona, é responsável por colocar e tirar os tijolos da prateleira de destino.

A movimentação das partes móveis da máquina é feita recorrendo a um variador de velocidade com controlo de posição. A posição atual destas é determinada recorrendo a *encoders* inseridos nos motores.

O processo de carga e descarga segue uma sequência pré determinada de movimentos que depende da posição atual e do estado de sensores colocados em cada parte móvel da máquina. O equipamento que leva e trás os tijolos da prateleira tem o nome de tela, e o equipamento que carrega e descarrega o material das prateleiras da vagona tem o nome de roleira, a qual possui uns garfos que entram dentro da vagona para auxiliar na movimentação dos tijolos para dentro desta.

A função gere a posição atual e posição de destino tanto da tela como da roleira, determina em que posição estão e controla o movimento de ambos fazendo com que este ocorra apenas quando as condições necessárias são cumpridas. Também gera alarmes caso alguma anomalia ocorra e permite a interligação com as funções de comunicação e controlo dos *drivers* de controlo dos motores.

Na figura 4.15 pode-se observar uma imagem onde é retratada a máquina e alguns dos seus movimentos.

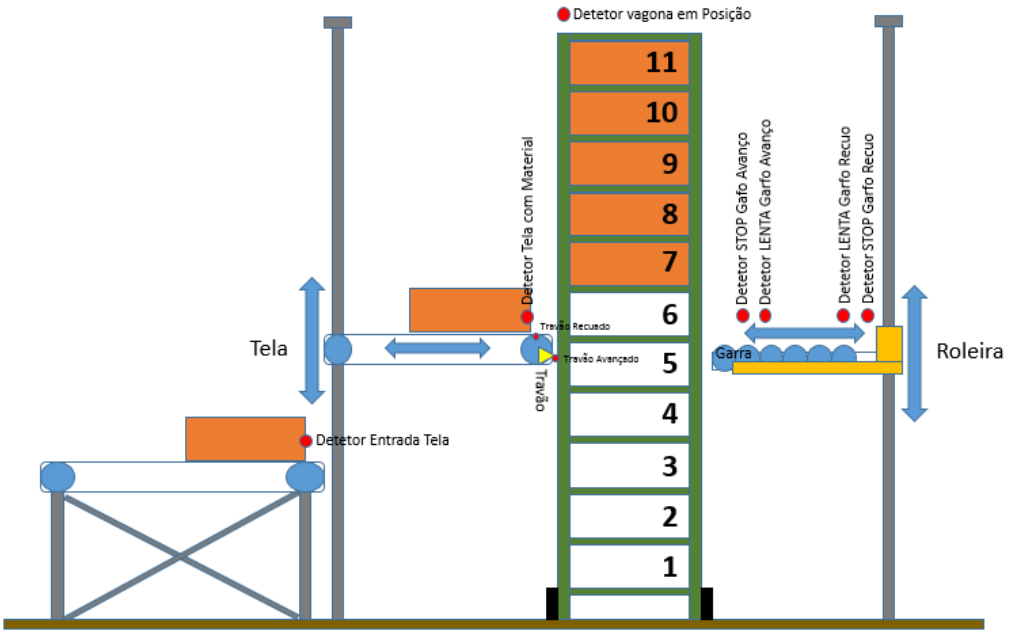


Figura 4.15: Ilustração de todos os componentes existentes na máquina de carga e descarga das vagonas do secador.

A ordem pela qual são percorridas as prateleiras das vagonas varia entre a carga e a descarga. Na carga as prateleiras são preenchidas no sentido descendente, e na descarga estas são esvaziadas no sentido ascendente da vagona.

De modo a facilitar a introdução no programa e a leitura das posições de cada uma das prateleiras, é necessária a criação de dois DB, um com as posições da tela e outro com as posições da roleira em relação a cada uma das prateleiras. Nestes DB são introduzidas e guardadas as posições do elevador da tela e do elevador da roleira. A função durante a sua execução acede a estes DB para determinar a próxima posição destino para ambos os equipamentos.

Nas entradas da função não consta nenhum parâmetro que diga se esta é para carga ou para descarga das vagonas, esta condição é determinada lendo e comparando as duas primeiras posições dos DB de posição. Se a segunda posição for maior que a primeira é para descarga pois o funcionamento da máquina é no sentido ascendente. Caso a segunda posição lida seja menor que a primeira, significa que se trata de uma carga, pois a sequência de carga é feita no sentido descendente.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Pela lista das saídas e entradas do FB será possível compreender grande parte do funcionamento desta.

- **Entradas**
  - *Enable* – Condição on/off da FB; (*Bool*)
  - *Automatic_Mode* – Identifica o modo de funcionamento da máquina, a 1 em automático e a 0 em manual; (*Bool*)

- *Number_Prat* – Número de prateleiras da vagona; (*Int*)
- *Cont_Man_Tela* – Byte de controlo manual do elevador da tela; (*Byte*)
- *Cont_Man_Roleira* – Byte de controlo manual do elevador da roleira; (*Byte*)
- *Pos_Act_Tela* – Posição atual do elevador da tela; (*Real*)
- *Pos_Act_Roleira* – Posição atual do elevador da roleira; (*Real*)
- *Det_Vag_In_Pos* – Detetor que sinaliza que a vagona se encontra em posição para carga ou descarga; (*Boo*l)
- *Det_Tela_With_Mat* – Detetor que indica que a tela se encontra com material; (*Bool*)
- *Det_Ent_Tela* – Detetor que indica que se encontra material para entrar na tela ou que é possível retirar o material desta; (*Bool*)
- *Brake_Advanced* – Indicação de que o travão responsável por manter a tela e a roleira alinhadas se encontra avançado; (*Bool*)
- *Brake_Recessed* – Indicação de que o travão responsável por manter a tela e a roleira alinhadas se encontra recuado; (*Bool*)
- *Detectors_Garfos* – Byte com o conjunto de detetores dos garfos; (*Byte*)
- *Vel_Ele* – Velocidade pretendida dos elevadores; (*Real*)
- *Vel_Adjust_Pos_Ele* – Velocidade de ajuste de posição do elevador da roleira; (*Real*)

A roleira quando se encontra com os garfos avançados, e ajusta a sua posição, a velocidade de movimento é muito mais lenta do que a de movimentação normal.

- *Vel_Tela* – Velocidade de movimentação da tela; (*Real*)
- *Vel_Roleira* – Velocidade de movimentação da roleira; (*Real*)
- *Vel_Garf_Rap* – Velocidade de movimentação rápida dos garfos; (*Real*)
- *Vel_Garf_Lent* – Velocidade de movimentação lenta dos garfos; (*Real*)

A movimentação rápida e lenta dos garfos depende da posição atual dos mesmos e do estado dos seus sensores.

- *DB_Pos_Ele_Tela* – DB com as posições do elevador da tela; (DB)
- *DB_Pos_Ele_Roleira* – DB com as posições do elevador da roleira; (DB)

Nas duas últimas entradas são indicadas os respetivos DB de posição. Estas duas entradas estão definidas como tendo um *data_type* de *Block_DB*.

- ***Byte* de controlo manual da tela**
  - Bit 0 – Subir elevador tela;
  - Bit 1 – Descer elevador tela;
  - Bit 2 – Mover tela no sentido CW;
  - Bit 3 – Mover tela no sentido CCW;
  - Bit 4 – Avançar travão;
  - Bit 5 – Recuar travão;

- ***Byte* de controlo manual da roleira**
  - Bit 0 – Subir elevador da roleira;
  - Bit 1 – Descer elevador da roleira;

- Bit 2 – Avançar garfos;
- Bit 3 – Recuar garfos;
- Bit 4 – Mover roleira no sentido CW;
- Bit 5 – Mover roleira no sentido CCW;

➢ ***Byte* contendo o estado dos detetores dos garfos**
- Bit 0 – Detetor de stop dos garfos recuados;
- Bit 1 – Detetor de lenta dos garfos recuados;
- Bit 2 – Detetor de stop dos garfos avançados;
- Bit 3 – Detetor de lenta dos garfos avançados;

➢ **Saídas**
- *Pos_Dest_Tela* – Posição de destino do elevador da tela; (*Real*)
- *Pos_Dest_Roleira* – Posição de destino do elevador da roleira; (*Real*)
- *Prat_Act_Tela* – Prateleira atual em que se encontra o elevador da tela; (*Int*)
- *Prat_Act_Roleira* – Prateleira atual em que se encontra o elevador da roleira; (*Int*)

As duas saídas anteriores existem para que seja possível colocar a prateleira em que os elevadores se encontram num sistema HMI.

- *Can_Move_Elev_Tela* – Indicação de que o elevador da tela se pode movimentar; (*Bool*)
- *Can_Move_Elev_Roleira* – Indicação de que o elevador da roleira se pode movimentar; (*Bool*)
- *Byte_Cont_Movimots* – Byte de controlo de funcionamento dos motores acionados por movimots existentes na máquina; (*Byte*)

Movimots são os inversores de frequência utilizados no acionamento de alguns motores da máquina.

- Vel_Ele_Tela – Velocidade do elevador da tela; (*Real*)
- Vel_Ele_Roleira – Velocidade do elevador da roleira; (*Real*)
- Vel_Mov_Tela – Velocidade de movimentação da tela do elevador da tela; (*Real*)
- Vel_Mov_Garfos – Velocidade de movimentação dos garfos da roleira; (*Real*)
- Vel_Mov_Roleira – Velocidade de movimentação das roleiras dos garfos; (*Real*)
- Liga_Travao – Avançar travão; (*Bool*)
- Desliga_Travao – Recuar travão; (*Bool*)
- Retirar_Vagona – Indicação para se retirar a vagona, esta ordem é dada quando a vagona está completamente cheia ou vazia, dependendo se é carga ou descarga; (*Bool*)

A pesquisa das posições de destino nos DB de posição é feita recorrendo a ponteiros para percorrer o *data block* até se encontrar a posição correta.

Em *Siemens*, não é possível utilizar ponteiros com a linguagem *Ladder*. Para ser possível programar ponteiros tem que se recorrer à linguagem STL. Um benefício que o *software* da

Siemens permite, é o de ser possível dentro da mesma função ter *networks* feitas em *Ladder* e *networks* feitas em STL. Isto permite criar as funções em linguagem *Ladder* e apenas recorrer ao STL quando estritamente necessário, visto que não se trata de uma linguagem muito fácil de aprender e compreender.

Na figura 4.16 é mostrado um pequeno excerto do código STL da função de carga e descarga das vagonas do secador.

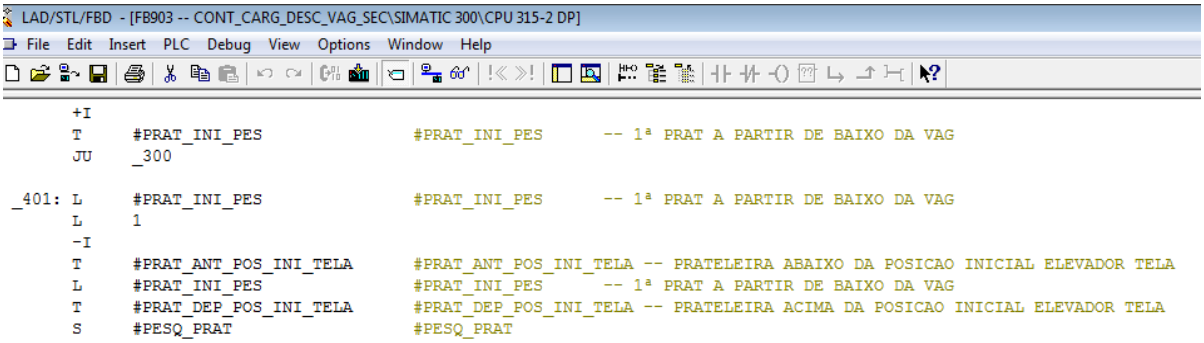


Figura 4.16: Excerto de código dos ponteiros que fazem a leitura das posições do DB de posições.

4.9. Controlo da Movimentação dos Carros dos Ventiladores do Secador

O secador semi-contínuo possui diversas linhas de vagonas no seu interior, entre essas linhas circulam os ventiladores que fazem circular o ar pelo secador. O movimento desses ventiladores em paralelo em relação às linhas das vagonas é feito recorrendo a um género de carro (figura 4.17) que transporta os ventiladores.

Figura 4.17: Imagem de um carro de ventilador do secador.

A função criada para o controlo destes carros permite controlar até um máximo de 8 linhas, mas o normal é sempre um número de linhas abaixo do máximo permitido pelo FB.

Os carros da mesma linha movimentam-se sempre no mesmo sentido e possuem zonas de passagem específicas, assim sendo, mesmo que uma linha possua 10 carros nenhum deles irá entrar no espaço de movimentação dos outros. Tendo isso em consideração é possível afirmar

que a função criada controla os diferentes tipos de movimentos das linhas e não o sentido de movimentação individual de cada carro. Existem sensores colocados nas linhas que indicam se os carros estão na posição de avanço ou recuo máximo.

Existem três modos de funcionamento possíveis (figura 4.18), modo síncrono, modo inverso e modo aleatório. O modo síncrono obriga todos os carros a movimentarem-se no mesmo sentido em todas as linhas e sempre que um carro chega a um dos detetores de avanço ou recuo espera que todos os carros cheguem a esse detetor e estejam prontos para inverter o movimento e só depois inverte o movimento.

O modo inverso alterna os sentidos de movimento de linha para linha e a inversão dos carros é feita da mesma forma do modo síncrono mas quando o carro atinge um dos detetores (avanço ou recuo máximo) espera pelos carros que se deslocam no mesmo sentido que ele, e ignora os que se deslocam no sentido oposto.

No modo aleatório os carros não olham ao sentido de movimento das outras linhas e quando vão inverter apenas esperam o tempo definido para inversão sem olhar ao estado dos outros carros.

Estes tempos de espera para inverter o sentido de movimentação existem para evitar fazer alterações muitos bruscas no sentido de movimento dos motores para prevenir a danificação do material.

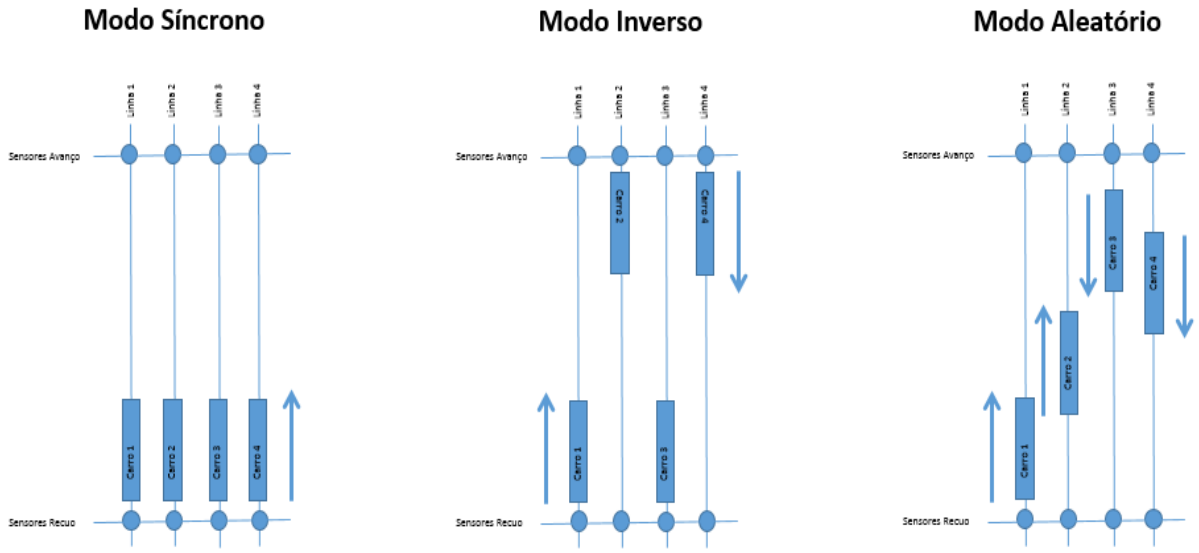


Figura 4.18: Ilustração dos três nodos de movimentação possíveis.

É definido um tempo máximo de duração da viagem dos carros entre sensores. Caso este tempo passe e o carro não atinja o sensor de fim de curso essa linha é parada, é gerado um alarme e as outras linhas continuam com a movimentação normal. Além deste tempo, também é introduzido um tempo na função que determina o tempo máximo que um carro pode ficar à espera dos outros nos detetores de fim de curso, tanto no movimento síncrono como no inverso, no movimento aleatório este tempo é ignorado. É gerado um aviso de sincronização caso o tempo de espera de um carro pelos outros for excedido.

Caso alguma das linhas seja desligada, as outras continuam com a movimentação normal. Em modo aleatório na altura do arranque das linhas, estas arrancam com algum tempo de atraso entre elas.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Os parâmetros de entrada e saída da função de controlo da movimentação dos carros dos ventiladores do secador são os seguintes.

- **Entradas**
  - *Enable* – Condição on/off da função; (*Bool*)
  - *Auto* – *Byte* de entrada da função onde cada bit identifica o modo de funcionamento automático de cada linha; (*Bool*)
  - *Num_lines* – Número de linhas do secador; (*Int*)
  - *Sync_Mode* – Modo de funcionamento síncrono; (*Bool*)
  - *Reve_Mode* – Modo de funcionamento inverso; (*Bool*)
  - *Rand_Mode* – Modo de funcionamento aleatório; (*Bool*)

  Se for ligado mais do que um modo de funcionamento ao mesmo tempo a função desliga as saídas e bloqueia o funcionamento dos motores.

  - *Can_Work* – Byte de entrada da função onde cada bit corresponde à condição de que cada linha pode trabalhar; (*Byte*)
  - *Det_Adv* – Byte de entrada da função onde cada bit corresponde ao detetor de avanço de cada linha; (*Byte*)
  - *Det_Rew* – Byte de entrada da função onde cada bit corresponde ao detetor de recuo de cada linha; (*Byte*)
  - *Reve_Time* – Tempo de paragem antes da inversão do movimento em segundos; (*Int*)
  - *Max_Time_Sinc* – Tempo máximo para sincronizar linhas em segundos; (*Int*)
  - *Max_Time_Mov* – Tempo máximo de duração do movimento entre sensores em segundos; (*Int*)
  - *Reset_Error* – *Reset* dos erros da função; (*Bool*)

- **Saídas**
  - *Adv_Carros* – Byte contendo o sinal individual de cada linha com a ordem para avançar os carros; (*Byte*)
  - *Rew_Carros* – Byte contendo o sinal individual de cada linha com a ordem para recuar os carros; (*Byte*)
  - *Error_Mov* – Byte contendo os bits de erro, que indica que o tempo máximo de movimentação de determinada linha foi excedido; (*Byte*)
  - *Warning_Sync* – Tempo máximo de sincronização excedido; (*Bool*)

4.10. Controlo dos Ventiladores Internos do Secador

Os carros referidos no ponto anterior servem para movimentar os ventiladores (figura 4.19) que fazem circular o ar no interior do secador.

Figura 4.19: Ventiladores internos do secador.

A função de controlo da ventilação do secador, ao contrário da anterior, apenas permite controlar os ventiladores de uma linha do secador, até um máximo de 16 carros de ventilação. De forma a ser possível controlar os ventiladores de todas as linhas e a respeitar e garantir as direções corretas de ventilação de todas, foi criado um parâmetro de entrada para interligar as várias linhas de ventilação utilizando um FB para cada uma.

Existem dois tipos de ventilação possível na função, ventilação paralela, ou ventilação alternada (figura 4.20). Na ventilação paralela todos os ventiladores de uma linha estão a rodar no mesmo sentido, por sua vez na ventilação alternada estes encontram-se com sentidos alternados entre eles.

Ventilação Paralela

L1	L2	L3	L4
→	→	→	→
→	→	→	→
→	→	→	→
→	→	→	→
→	→	→	→
→	→	→	→
→	→	→	→
→	→	→	→
→	→	→	→

Ventilação alternada

L1	L2	L3	L4
→	→	→	→
←	←	←	←
→	→	→	→
←	←	←	←
→	→	→	→
←	←	←	←
→	→	→	→
←	←	←	←
→	→	→	→

Figura 4.20: Ilustração dos tipos de ventilação possíveis com a função de controlo dos ventiladores internos do secador.

O tipo de ventilação é escolhido recorrendo a uma entrada da função, e pode ser alterado a qualquer altura, fazendo com que os ventiladores da linha parem, esperem algum tempo e arranquem com o novo tipo de ventilação.

O sentido dos ventiladores na linha também pode ser invertido ou não. Se for invertível cada linha espera que a linha anterior acabe a inversão ou que esteja desligada ou que esteja sem inversão e só depois começa o seu processo de inversão. A inversão de cada linha é feita ventilador a ventilador (figura 4.21), onde é enviada ordem de paragem para o ventilador, aguarda-se algum tempo para garantir que ele parou e envia-se a ordem de arranque no sentido oposto.

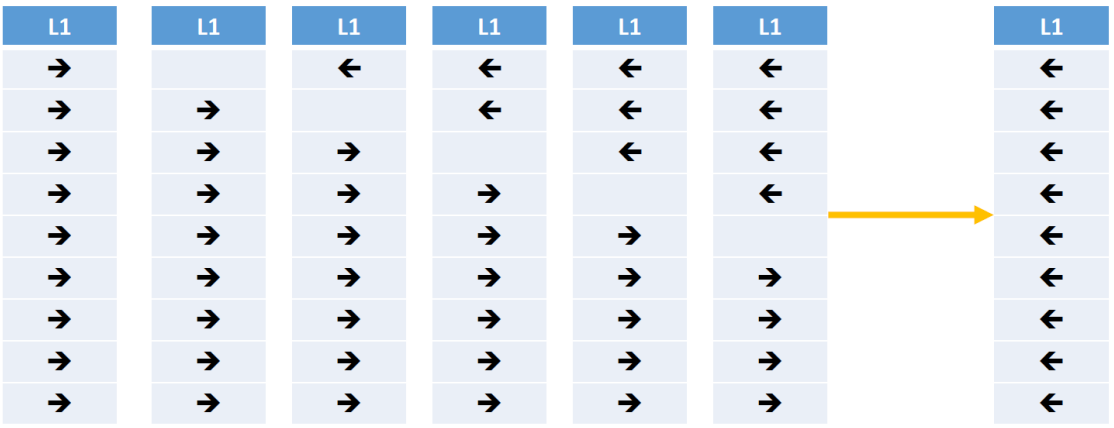


Figura 4.21: Ilustração da sequência de inversão do sentido de ventilação de uma linha do secador.

No momento que a linha termina a inversão do sentido de ventilação é colocada uma saída da função a 1 que indica a conclusão da inversão nessa linha e a autorização para iniciar a inversão da próxima.

Quando se desliga uma linha ou se altera o seu tipo ou sentido de ventilação, esta arranca no sentido certo tendo em conta o resto das linhas, para tal existe um parâmetro de entrada da FB que indica se o ventilador se encontra no sentido inicial ou invertido, este parâmetro tem que ser alternado entre 0 e 1 sempre que termine a inversão de todas as linhas do secador.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Após concluída a função, esta ficou com o seguinte aspeto em termos de entradas e saídas.

- **Entradas**
  - *Enable* – Condição on/off da FB de controlo dos ventiladores; (*Bool*)
  - *Num_Cars* – Número de carros existentes na linha (máximo de 16); (*Int*)
  - *Inv_Cycle* – Tempo de espera antes de iniciar a inversão (0 = sem inversão, 1 - 120 minutos); (*Time*)
  - *Stop_Time* – Tempo de paragem a quando da inversão (150 - 300 segundos); (*Time*)
  - *Start_Time* – Tempo de arranque dos ventiladores; (*Time*)

Durante o arranque na linha, de modo a evitar excesso de carga na rede, os ventiladores arrancam sequencialmente. A entrada da função Start_Time é o tempo de intervalo entre arranques.

- *Initial_Direction* – Sentido inicial da linha (0=CW, 1= CCW); (*Bool*)
- *Inv_Prev_Line* – Indicação de que a linha anterior terminou a inversão; (*Bool*)
- *Type_Ventilation* – Tipo de ventilação (0=Vent_Par, 1=Vent_Alt); (*Bool*)
- *Current_Direction* – Sentido atual da ventilação do secador; (*Bool*)

➢ **Saídas**

- *Finished_Inv* – Indicação de que terminou a inversão da linha; (*Bool*)
- *Left_Vent* – Ordem para os ventiladores rodarem para a esquerda, esta saída é to tipo *word*, em que cada bit corresponde a um ventilador; (*Word*)
- *Right_Vent* – Ordem para os ventiladores rodarem para a direita, esta saída é to tipo *Word*, em que cada bit corresponde a um ventilador; (*Word*)

No arranque da ventilação do secador é necessário criar condições para que a entrada "*Inv_Prev_Line*" da função que controla a ventilação da primeira linha seja posta a um. Esta condição força o início do processo de inversão da ventilação do secador (figura 4.22).

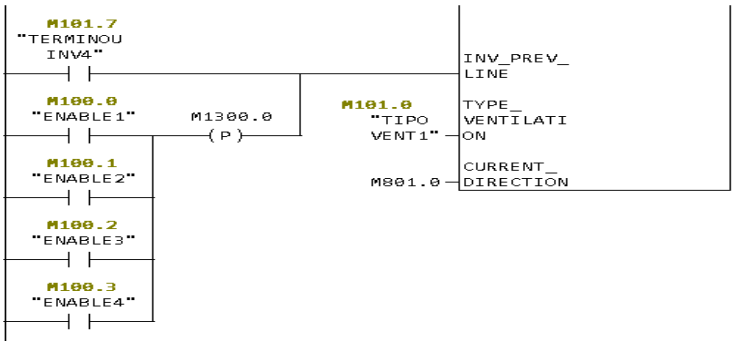


Figura 4.22: Excerto de código que mostra algumas entradas da função de controlo da ventilação interna do secador, e as condições de inversão do sentido de ventilação.

4.11. Controlo do Posicionamento do Carro das Vagonas do Forno

O tijolo após sair do secador, é colocado noutro tipo de vagonas para poder entrar no forno. Estas vagonas circulam em linhas paralelas entre si e ao forno. As vagonas podem permanecer nestas linhas, com material enquanto esperam para entrar no forno, ou sem material à espera de receber mais material

A movimentação das vagonas nas linhas é assegurada por um carro (figura 4.23) que percorre a linha de uma ponta à outra puxado por cabos de aço. Este carro possui um "dente" que quando passa pelo batente existente na vagona sobe e permite puxar a mesma até à posição desejada. O carro quando recua vai buscar a vagona e quando avança trás a vagona.

Figura 4.23: Imagem de um carro a puxar uma vagona.

Cada linha possui um conjunto de sensores que auxiliam o carro a posicionar-se, tais como a indicação de vagona no início da linha, vagona no fim da linha, linha cheia, avanço e recuo máximo do carro (figura 4.24).

O carro possui um conjunto pré definido de movimentos que tem que cumprir para assegurar que trouxe a vagona para o sítio certo, sempre que vai buscar uma vagona. O carro para iniciar esse conjunto de movimentos tem que se encontrar junto ao sensor de avanço máximo do carro, onde a sua posição atual é forçada para um valor pré definido. A posição de destino do carro é calculada tendo em conta o número de vagonas na linha, o tamanho da vagona e uns valores de *offset* que são introduzidos na entrada da função.

O carro após chegar à posição de destino começa o movimento de avanço trazendo a vagona. Ele apenas para este movimento quando ativa o sensor de avanço máximo do carro. Após este movimento, é calculada uma posição de recuo suficiente para que o carro consiga agarrar a vagona por trás e empurrá-la para o fim da linha, ficando depois à espera que esta seja retirada. Após a saída da vagona da linha o carro retorna ao avanço máximo do carro e reinicia a sequência de movimentos caso haja mais vagonas para ir buscar.

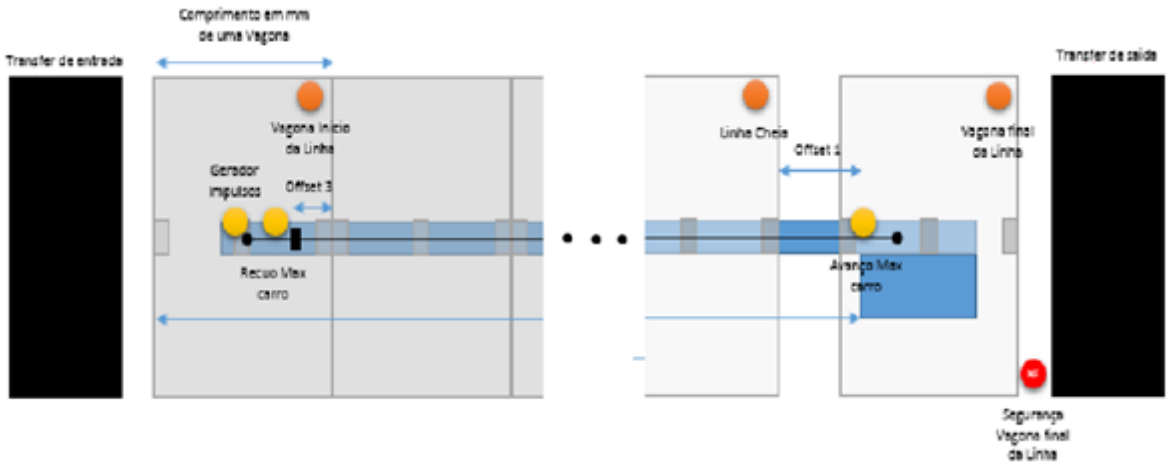


Figura 4.24: Esquema representativo de todos os sensores existentes numa linha de vagonas.

A contagem das vagonas presentes na linha é feita recorrendo à sequências de estados dos sensores tanto de entrada como de saída da linha. Sempre que a sequência dos sensores de entrada ocorre é adicionado 1 ao número de vagonas na linha. Sempre que a sequência dos sensores de saída ocorre é subtraído 1 ao número de vagonas na linha. Estas operações permitem saber o número de vagonas presente na linha a qualquer momento. Também é possível alterar este valor na entrada da função, caso tenha havido algum problema com os sensores e as

contagens estejam erradas, bastando para isso alterar o parâmetro correspondente ao número de vagonas na linha dentro da DB de instância da função.

A distância percorrida pelo carro é calculada recorrendo a impulsos dados por um gerador de impulsos colocado junto do enrolador do cabo do carro. Existe um parâmetro de entrada da FB onde é colocada a distância em milímetros por impulso.

A função também permite a deteção e em alguns casos a correção de vários erros, nomeadamente, erros no contador de impulsos, erro do número de vagonas trazidas pelo carro ser superior a uma, ou não trazer vagona, erros de estado de sensores, erros de contagem de vagonas, etc.

A inversão de sentido de rotação do motor também foi tida em consideração, para evitar inversões bruscas foi criado um *timer* que apenas deixa inverter o sentido do motor quando passa o tempo definido para ele.

Outra situação tida em consideração durante a criação da função foi a da entrada de vagona na linha enquanto o carro se encontra a ir buscar uma vagona. Nestas situações e se a distância do carro à vagona o permitir, o carro para, espera que a vagona acabe de entrar e recalcula a posição de destino tendo em consideração o acréscimo da vagona que acabou de entrar.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Após concluída a função criada para o controlo deste tipo de carros ficou com as seguintes entradas e saídas.

- **Entradas**
  - *Enable* – Condição on/off da FB de controlo do carro; (*Bool*)
  - *Automatic_Mode* – Modo de funcionamento automático quando colocada a 1 e em manual quando a 0; (*Bool*)
  - *Num_Max_Vag* – Número máximo de vagonas que a linha suporta; (*Int*)
  - *Vag_Size* – Tamanho da vagona (mm); (*Real*)
  - *mm_Pulse* – A quantos milímetros corresponde um impulso (mm); (*Real*)
  - *Offset_1* – Offset 1 (mm), distância entre o avanço máximo do carro e o sensor de linha cheia; (*Real*)
  - *Offset_2* – Offset 2 (mm), distância entre o avanço máximo do carro e o início da linha; (*Real*)
  - *Offset_3* – Offset 3 (mm) distância entre o batente da vagona e margem de recuo do carro; (*Real*)
  - *Det_Ava_Max_Car* – Detetor de avanço máximo do carro; (*Bool*)
  - *Det_Rec_Max_Car* – Detetor de recuo máximo do carro; (*Bool*)
  - *Ent_New_Vag* – Indicação que está a entrar nova vagona na linha;
  - *Exit_Vag* – Indicação que está a sair uma vagona da linha; (*Bool*)
  - *Det_Vag_Ini_Line* – Detetor de vagona no início da linha; (*Bool*)
  - *Det_Vag_End_Line* – Detetor de vagona no fim da linha; (*Bool*)
  - *Det_Vag_End_Line_Seg* – Detetor de segurança de vagona no final da linha; (*Bool*)
  - *Det_Vag_Line_Filled* – Detetor de linha cheia; (*Bool*)

- *Pulse_Generator* – Detetor do gerador de impulsos; (*Bool*)
- *Ava_Car_Man* – Avanço manual do carro; (*Bool*)
- *Rec_Car_Man* – Recuo manual do carro; (*Bool*)
- *Reset* – *Reset* dos erros da função; (*Bool*)

➢ **Saídas**

- *Ava* – Avança o carro; (*Bool*)
- *Rec* – Recua o carro; (*Bool*)
- *Error_Pulse_Generator* – Erro no gerador de impulsos; (*Bool*)
- *Error_Cont_Vag* – Erro de contagem de vagonas; (*Bool*)

Dentro do DB de instância da função é possível encontrar as seguintes variáveis.

➢ **DB**

- *Pos_Carro* – Posição atual do carro em milímetros; (*Real*)
- *Pos_Destino* – Posição de destino do carro em milímetros; (*Real*)
- *Num_Vag_Linha* – Número de vagonas na linha; (*Int*)

4.12. Ocupação das Linhas

De forma a auxiliar o operador e a ser possível visualizar na supervisão e em aparelhos HMI a forma como a linha está ocupada e por que tipo de material, foi criada uma função para esse efeito.

A função no seu DB de instância possui um vetor de inteiros, em que cada valor representa uma vagona na linha. Se esses valores forem zero quer dizer que o lugar não se encontra ocupado por nenhuma vagona, se for diferente de zero representa que se encontra uma vagona em determinada posição e o tipo de material que se encontra nela. Cada tipo de material tem um código associado que é mostrado aos operadores quando eles solicitam tal informação (figura 4.25).

Figura 4.25: Representação da distribuição das vagonas em uma linha.

A forma como a contagem, a saída e a entrada das vagonas na linha está programada é idêntica à função anteriormente descrita.

A função permite a visualização de linhas com um máximo de 52 vagonas, e caso surjam erros na disposição das vagonas existe uma entrada que quando colocada a 1 permite ao operador a modificação do vetor interno representativo da linha.

Outra funcionalidade implementada é a de ser possível observar se existe alguma vagona na posição de saída da linha ou não, quando uma vagona se desloca para essa posição, a função modifica o vetor de forma a este mostrar que uma vagona se encontra para sair da linha e o tipo de material que ela transporta.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

As entradas e saídas da função ficaram com as seguintes variáveis.

- **Entradas**
  - *Imput_Vag* – Indicação de entrada de vagona; (*Bool*)
  - *Output_Vag* – Indicação de saída de vagona; (*Bool*)
  - *Vag_End_Line* – Sensor de vagona no fim da linha; (*Bool*)
  - *Vag_Begi_Line* – Sensor de vagona no início da linha; (*Bool*)
  - *Type_Of_Mat* – Tipo de material na próxima vagona a entrar na linha; (*Int*)
  - *Num_Max_Vag* – Número máximo de vagonas que a linha suporta; (*Int*)
  - *Change_Layout_Vag* – Alterar disposição do posicionamento das vagonas na linha; (*Bool*)

O *array* contendo a disposição da linha e o tipo de material pode ser encontrado dentro da DB de instância da função.

- **DB**
  - *Pos_Vag* [1..52] – Vetor de visualização da linha; (*Array*)

- **Saídas**
  - *Error_Cont_Vag* – Erro na contagem das vagonas; (*Bool*)
  - *Number_Of_Vag* – Número de vagonas presentes na linha; (*Bool*)

4.13. Controlo da Translação do *Transfer*

O *transfer* (figura 4.26) é o equipamento responsável por transportar as vagonas de umas linhas para outras, para a entrada do forno e da saída do forno para as linhas de embalamento. Ele movimenta-se de forma perpendicular às linhas nos topos das mesmas, e pode movimentar-se entra várias linhas.

A função criada permite controlar o movimento do *transfer*, alterar a sua velocidade, determinar se chegou à linha de destino e alinhá-lo com as linhas de forma a permitir a transposição das vagonas das linhas para cima do *transfer* e vice-versa.

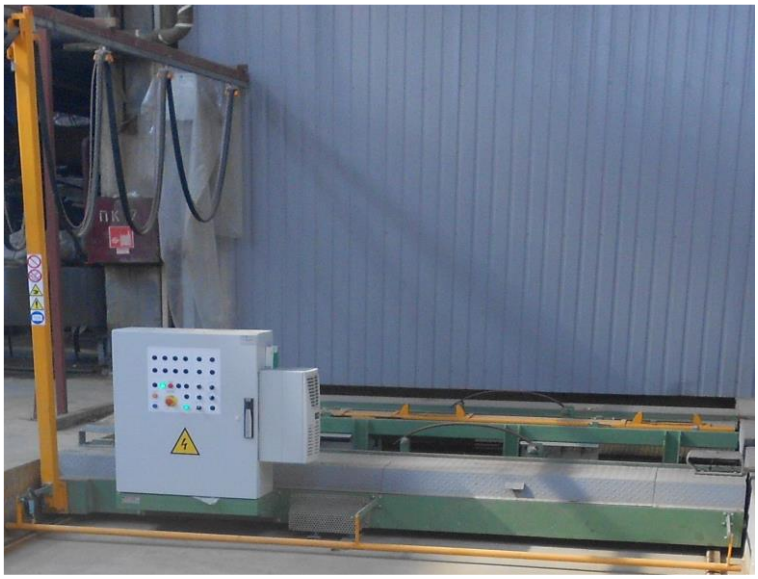


Figura 4.26: Imagem de um *transfer* de vagonas.

O *transfer* sempre que se encontra a chegar à linha de destino passa a movimentar-se com uma velocidade mais lenta devido ao facto de existirem sensores que indicam que se está a aproximar de uma linha, e na eventualidade de essa linha ser a de destino ele passa para a velocidade lenta. Existe outro conjunto de sensores que indica ao *transfer* que ele está na posição certa.

A determinação da posição atual do *transfer* é possível recorrendo a um processo de contagem (figura 4.27) que sempre que ele passa por um dos sensores acima descritos incrementa ou decrementa um valor dependendo do sentido de movimentação. Na imagem seguinte é demonstrado o processo de contagem descrito.

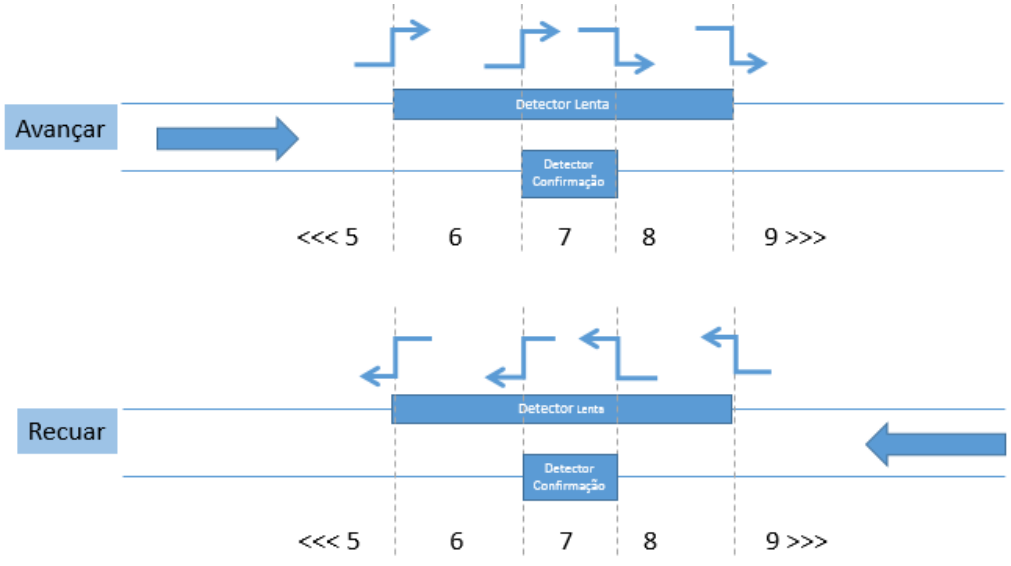


Figura 4.27: Condições de incremento e decremento para o posicionamento do *transfer*.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Quando controlado em modo manual o *transfer* abranda e para em todas as linhas. Em modo automático apenas abranda e para na linha de destino.

- **Entradas**
  - *Enable* – Condição on/off da FB de controlo do *transfer*; (*Bool*)
  - *Automatic_Mode* – Esta entrada quando colocada a 1 coloca o modo de funcionamento em automático, e quando colocada a 0 em manual; (*Bool*)
  - *Can_Work* – Indicação de que o *transfer* se pode movimentar, caso esteja a 0, desliga as saídas de movimentação da FB; (*Bool*)
  - *Man_Ava* – Avançar *transfer* em manual; (*Bool*)
  - *Man_Rec* – Recuar *transfer* em manual; (*Bool*)
  - *Number_of_Lines* – Número de linhas abrangidas pelo *transfer*, até um máximo de 16 linhas; (*Real*)
  - *Dest_Line* – Linha de destino do *transfer*; (*Real*)
  - *Det_Conf* – Sensor de confirmação do posicionamento do *transfer*; (*Bool*)

  O detetor de confirmação é único para o conjunto de todas as linhas. O que permite saber a linha em que o *transfer* se encontra é o detetor de lenta ativo no momento.

  - *Det_Security* – Sensor de segurança de movimentação do *transfer*; (*Bool*)
  - *Time_of_Adjustment* – Tempo de atraso para permitir a centragem do *transfer* com a linha durante a paragem e filtragem de erros, em milissegundos; (*Int*)
  - *Time_Stop_Man* – Tempo que o *transfer* para nas linhas e espera antes de retomar a marcha, em modo de funcionamento manual; (*Int*)
  - *Reset_Error* – *Reset* dos erros da função; (*Bool*)

Os sinais dos detetores de lenta foram colocados no interior do DB de instância da função. Como é possível à função controlar um *transfer* que percorre até 16 linhas é necessária uma *Word* para acomodar todos os bits de lenta existentes.

Esta Word pode ser encontrada a partir do byte 100 desta mesma DB.

- **DB**
  - *Detetor_Lenta* – Word contendo todos os detetores de lenta; (*Word*)

É possível aceder a esta variável utilizando a seguinte forma de endereçamento – DBx.DBW100, em que o x representa o número da DB em causa.

- **Saídas**
  - *Ava* – Sinal de ordem de avanço do *transfer*; (*Bool*)
  - *Rec* – Sinal de ordem de recuo do *transfer*; (*Bool*)
  - *Slow* – Indicação para o *transfer* mudar para velocidade lenta; (*Bool*)
  - *Trans_in_Pos* – *Transfer* encontra-se na posição pretendida; (*Bool*)
  - *Pos_Trans_Line* – Word contendo os bits que assinalam se o *transfer* se encontra numa linha e se sim em que linha é que ele se encontra; (*Word*)
  - *Error* – Erro de posição do *transfer*. Este encontra-se numa posição que não está a ser correspondida pelo estado dos sensores de confirmação e lenta; (*Bool*)

4.14. Controlo do Empurrador do *Transfer*

Inserido no *transfer* existe um sistema hidráulico cuja função é a de colocar ou retirar as vagonas de cima do *transfer* (figura 4.28).

Figura 4.28: Imagem de um empurrador de *transfer*.

O empurrador, por se tratar de um sistema hidráulico, possui uma centralina responsável por controlar o movimento do empurrador.

O FB descrito neste capítulo também é responsável pelo controlo do posicionador do *transfer*, este posicionador quando avançado bloqueia o movimento do *transfer*.

Dependendo se está a carregar ou a descarregar a vagona, o empurrador possui um conjunto de movimentos pré-definidos constituídos por um determinado número de recuos e avanços do empurrador.

A sequência de movimentos está descrita na figura 4.29.

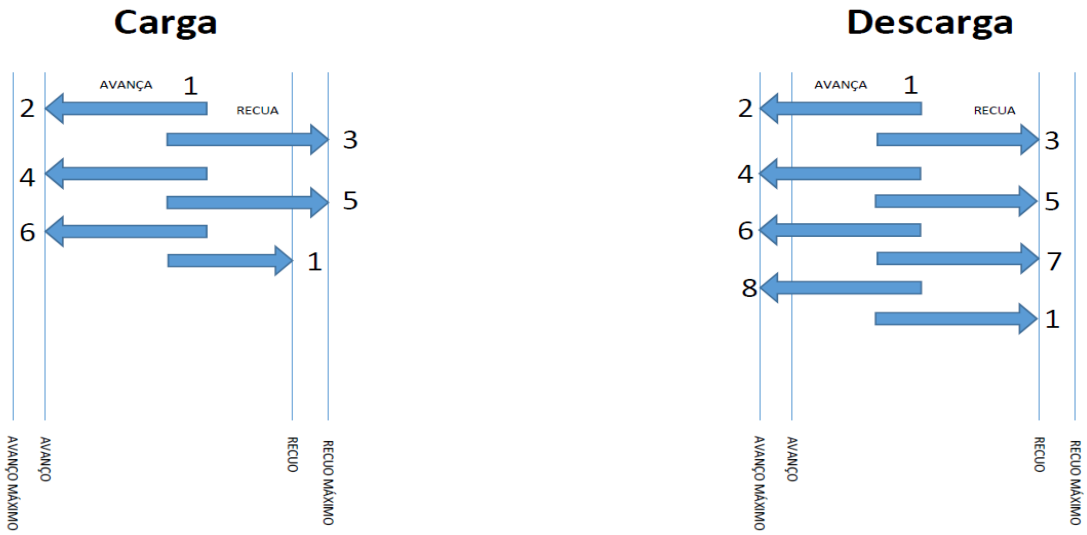


Figura 4.29: Sequências de movimentos do empurrador para a carga e a descarga.

Existe no total um conjunto de 4 sensores, dois em cada extremidade. As sequências de movimento são feitas alternando a posição do empurrador entre estes sensores, como se pode visualizar na figura 4.29.

A função não permite que se faça a manobra de descarga sem que exista vagona em cima do *transfer*, ou que se faça manobra de carga sem que exista vagona em espera para ser carregada no *transfer*.

O *grafcet* de nível 1 desta função encontra-se anexo a este relatório (Anexo B), o *grafcet* demonstra o funcionamento do movimento do empurrado.

Quando existe ordem de avanço no empurrador é seguida uma série de ordens dadas à centralina do empurrador.

- Avançar posicionador
  - Ligar motor;
  - +2s - Ligar válvula de alívio;
  - Liga válvula de avanço;

  Quando chega à posição de avanço

  - Desligar válvulas;
  - +5s – desligar motor;

O mesmo acontece quando existe ordem de recuo do empurrador.

- Recuar posicionador
  - Ligar motor;
  - +2s - Ligar válvula de alívio;
  - Liga válvula de recuo;

  Quando chega à posição de recuo

  - Desligar válvulas
  - +5s – desligar motor

Estes dois conjuntos de instruções dadas à centralina são gerados e controlados dentro do FB de controlo do empurrador.

A função quando concluída ficou com o seguinte esquema de entradas e saídas.

- **Entradas**
  - *Enable* – Condição on/off da FB de controlo do empurrador; (*Bool*)
  - *Automatic_Mode* – Esta entrada quando colocada a 1 coloca o modo de funcionamento em automático e quando colocada a 0 em manual; (*Bool*)
  - *Trans_in_Pos* – Indicação de que o *transfer* se encontra em posição; (*Bool*)
  - *Rec_Emp_Man* – Recuar empurrador em manual; (*Bool*)
  - *Ava_Emp_Man* – Avançar empurrador em manual; (*Bool*)
  - *Rec_Pos_Man* – Recua posicionador em manual; (*Bool*)
  - *Ava_Pos_Man* – Avança posicionador em manual; (*Bool*)
  - *Load_Cycle* – Ordem para carregar vagona para o *transfer*; (*Bool*)
  - *Discharge_Cycle* – Ordem para descarregar a vagona do *transfer*; (*Bool*)
  - *Trans_With_Vag* – Indicação de que o *transfer* se encontra com vagona; (*Bool*)
  - *Det_Ava_Emp* – Detetor de empurrador no avanço; (*Bool*)
  - *Det_Ava_Max_Emp* – Detetor de empurrador no avanço máximo; (*Bool*)
  - *Det_Rec_Emp* – Detetor de empurrador no recuo; (*Bool*)
  - *Det_Rec_Max_Emp* – Detetor de empurrador no recuo máximo; (*Bool*)
  - *Det_Ava_Pos* – Detetor de posicionador avançado; (*Bool*)

- *Det_Rec_Pos* – Detetor de posicionador recuado; (*Bool*)
- *Reset_Error* – *Reset* aos erros da função; (*Bool*)

➢ **Saídas**

- *Cyclle_Complete* – Indicação de conclusão do ciclo de carga e descarga; (*Bool*)
- *On_Off_Centralina* – Controlo on/off da centralina; (*Bool*)
- *Val_Cent_Relief* – Válvula de alívio da centralina; (*Bool*)
- *Val_Cent_Ava_Pos* – Válvula de avanço do posicionador; (*Bool*)
- *Val_Cent_Rec_Pos* – Válvula de recuo do posicionador; (*Bool*)
- *Val_Cent_Ava_Emp* – Válvula de avanço do empurrador; (*Bool*)
- *Val_Cent_Rec_Emp* – Válvula de recuo do empurrador; (*Bool*)
- *Error_Mov* – Erro de movimentação do empurrador; (*Bool*)

4.15. Controlo dos Queimadores Pulsados do Forno

Um forno de túnel para cerâmica ao longo do seu comprimento pode conter vários queimadores pulsados (figura 4.30). A função criada serve para controlar a abertura e fecho das válvulas de combustível de cada um destes queimadores. Ela, a função, permite até um máximo de 32 bicos por queimador.

Figura 4.30: Queimadores pulsados colocados em cima de um forno.

Para o controlo do funcionamento do queimador é utilizado um controlador PID. O controlador usado é uma função da *Siemens* que faz parte das bibliotecas do *Simatic Manager* (FB 41 – CONT_C). A função do controlador tem que ser chamada dentro de um OB de interrupção com um ciclo fixo entre interrupções (OB 35 por exemplo).

Uma vez que o FB de controlo do queimador é chamado dentro do OB 1 foi feito endereçamento indireto desta para o DB de instância do PID associado à função (figura 4.31). Este tipo de endereçamento permite ler e escrever valores no PID.

As válvulas de cada bico do queimador são controladas recorrendo a uma *DWord* contendo 32 bits, cada bit representa um bico do queimador.

O FB permite definir potências individuais, de 0 a 100%, para cada bico e uma potência geral máxima do queimador, também de 0 a 100%.

O tempo de abertura de cada bico ao longo de um minuto é definido tendo em conta o número de ciclos do queimador por minuto, o valor de saída do PID e a potência individual de cada bico, utilizando a seguinte fórmula para o efeito.

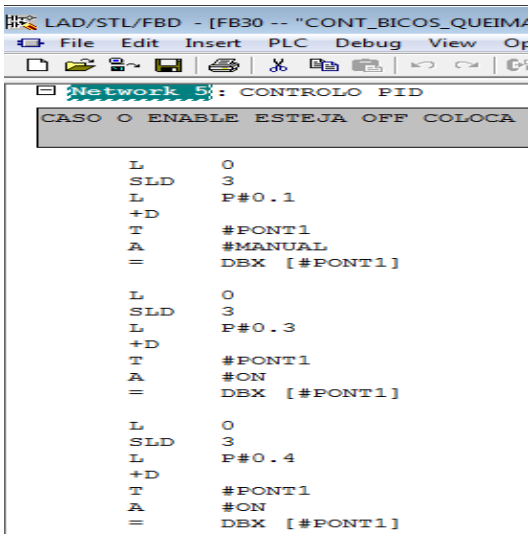


Figura 4.31: Excerto de código em STL da função de controlo dos queimadores pulsados.

$$\text{Tempo de abertura do bico } 1 = \frac{60}{Num\_Ciclos} \times Pot\_Out\_PID \times Pot\_Ind\_Bic1 \qquad (1)$$

No interior do FB existe uma condição de *shift* que percorre a *DWord* dos bicos em intervalos de tempo calculados pela equação (2) e que testa o estado dos bicos e o atualiza. Quanto maior for o número de ciclos por minuto mais rápido será o *shift* entre os bicos.

O tempo de *shift* é calculado da seguinte forma:

$$\text{Tempo de SHIFT entre bicos} = \frac{60}{\frac{Num\_Ciclos}{Num\_Bicos}} \qquad (2)$$

O controlador PID, em função do *setpoint* de temperatura introduzido e da temperatura medida, vai aumentar ou diminuir a potência do queimador, que se traduz em mais ou menos tempo de injeção de combustível por parte dos bicos.

Para evitar que as válvulas se danifiquem muito rapidamente, existe um parâmetro de entrada do FB onde é introduzido o tempo mínimo de aberturas das válvulas, caso o tempo calculado para a abertura da válvula seja inferior a este é forçada a abertura com o tempo mínimo definido.

Outro dos parâmetros de entrada da função que vale a pena salientar é o *offset* do DB de instância do PID. Por exemplo se o DB de instância da função do queimador for o DB 100, deve ser colocado no respetivo PID um DB de instância igual a 100 mais o *offset* introduzido na entrada da função. Com um *offset* de 10, o DB de instância do PID seria o DB 110.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

Após concluída, a função ficou com as seguintes entradas e saídas.

- **Entradas**
  - Enable – Condição on/off do FB de controlo do queimador; (*Bool*)
  - Manual – Colocar o PID em manual utilizando a potência definida para manual; (*Bool*)
  - Temp_Real – Temperatura real do queimador; (*Real*)
  - Temp_Ideal – Temperatura desejada para o queimador; (*Real*)
  - Pot_Maxima – Limite máximo de potência à saída do PID; (*Real*)
  - Pot_Minima – Limite mínimo de potência à saída do PID; (*Real*)
  - Pot_Manual – Valor da potência quando o PID se encontra em manual; (*Real*)
  - PID_P – Componente proporcional do PID; (*Real*)
  - PID_I – Componente integrativa do PID; (*Time*)
  - PID_D – Componente derivativa do PID; (*Time*)
  - Num_Bicos – Número de bicos existentes no queimador; (*Int*)
  - Num_Ciclos – Número de ciclos por minuto desejado; (*Real*)
  - Tempo_Min – Tempo mínimo de abertura das válvulas dos bicos em milissegundos; (*In*t)
  - *Offset_DB_PID* – Offset da DB de instância do PID associado ao queimador; (*Int*)

- **Entradas/Saídas**
  - Valvulas – *Dword* de controlo das válvulas do queimador; (*DWord*)

Esta variável é do tipo IN/OUT porque é necessário aceder ao estado atual das válvulas dos bicos, e se a variável fosse só de saída isso não seria possível.

- **Saídas**
  - Val_Gas_Queim – Válvula geral do combustível do queimador; (*Bool*)

Se os bicos estiverem todos desligados mais do que alguns segundos é desligada a válvula geral do combustível.

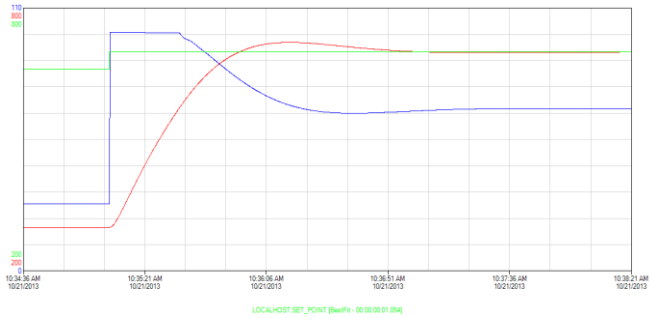
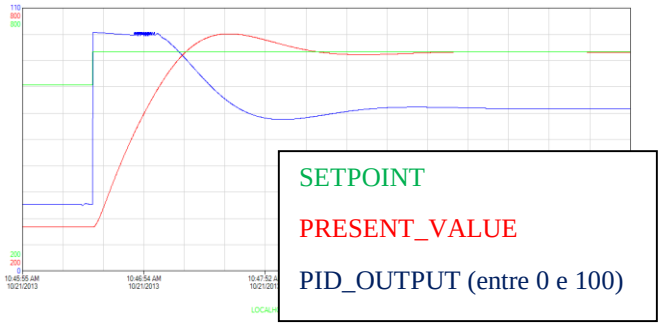
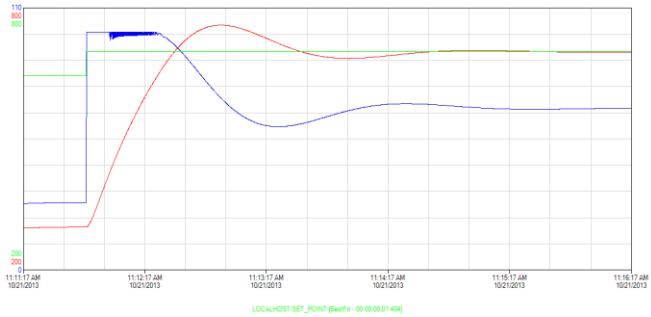
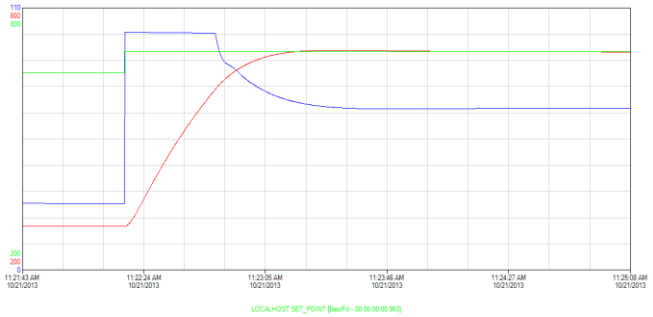
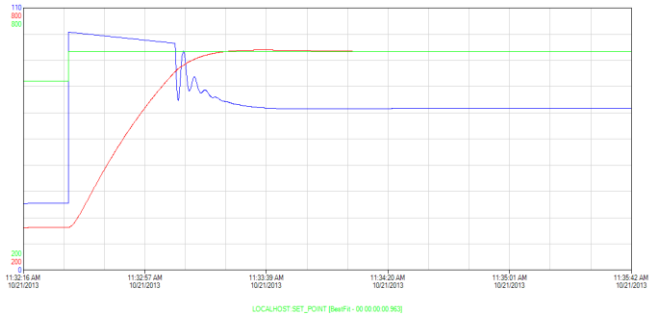
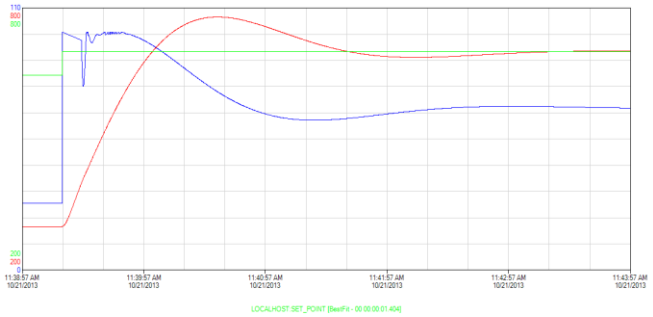
Por fim falta referir que as potências individuais de cada bico se encontram acessíveis dentro do DB de instância da função na zona das variáveis estáticas, na forma de um *array*.

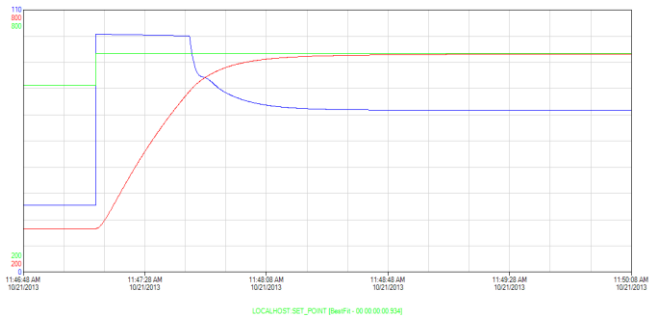
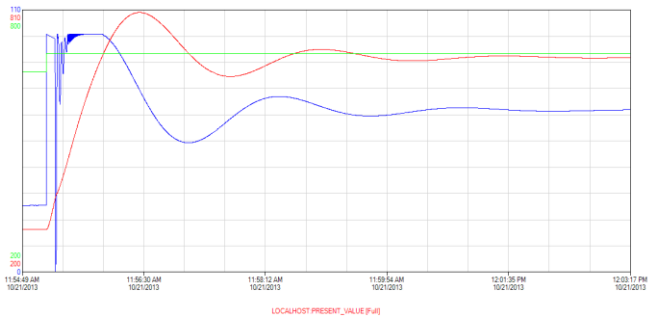
- **DB**
  - QP_Pot_Bic [0..31] – Vetor com a potência individual de cada bico do queimador. (*Array*)

Durante os testes feitos à função de controlo dos queimadores pulsados, e de forma a perceber-se melhor o comportamento do PID para diferentes valores de P, I e D, foram realizados alguns ensaios por simulação. Estes ensaios serviram para se observar as diferentes conjugações de valores de P, I e D e o seu efeito na resposta do controlador PID fornecido pela *Siemens.* Na

tabela seguinte (tabela 4.1) pode-se observar alguns dos resultados obtidos em termos de curvas de saída do PID com um tempo de amostragem de 100 milissegundos.

Tabela 4.1: Resposta do PID para várias conjugações de parâmetros P,I e D.

P=2 (Real)	**I=20s (time)**	**D=10s (time)**	**P=2,4 (Real)**	**I=16s (time)**	**D=15s (time)**
					
P=2,88 (Real)	**I=12s800ms (time)**	**D=22s500ms (time)**	**P=1,6 (Real)**	**I=24s (time)**	**D=5s (time)**
					
P=8 (Real)	**I=24s (time)**	**D=5s (time)**	**P=1,6 (Real)**	**I=24s (time)**	**D=25s (time)**
					

P=1,6 (Real)	I=48s (time)	D=5s (time)	P=2 (Real)	I=10s (time)	D=50s (time)
					

Como é possível ver pelos gráficos, para diferentes valores de P, I e D, o controlador assume diferentes comportamentos. Com o auxílio destes gráficos pode-se escolher os valores que melhor se adaptam à resposta que se deseja. Caso se deseje fazer uma otimização dos parâmetros do PID pode-se recorrer ao *MatLab* para simular o sistema de forma a se conseguir a melhor resposta possível para o sistema em causa.

4.16. FC de Ligação ao Variador LTP-B da SEW

A grande maioria das máquinas descritas até agora possui variadores eletrónicos de velocidade a acionar os motores. De forma a facilitar a programação da interface entre o variador e o autómato, foram criadas três funções do tipo FC (*function*) que gerem as ligações entre o autómato e o variador para três tipos diferentes de variador da marca SEW [8].

O primeiro desses três tipos de variador é o LTP-B. Este variador permite comunicação por *profibus*. A comunicação entre o PLC e o variador é feita recorrendo a um conjunto de *words*, três *words* de controlo e três *words* de estado, num total de 6 *words* (figura 4.32).

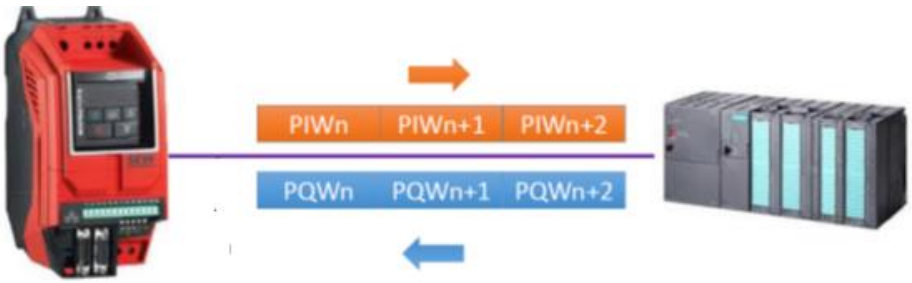


Figura 4.32: Ilustração das *words* de comunicação entre o LTP-B e um PLC *Siemens*.

As *words* de controlo enviadas do PLC para o variador são constituídas por *Control Word*, velocidade pretendida e por rampa de aceleração e desaceleração.

As *words* de estado enviadas do variador para o PLC são a *status word*, a velocidade atual e o consumo atual do motor.

Estas *words* são lidas e escritas de e para o variador utilizando endereçamento indireto. Este endereçamento é feito dizendo à função qual é o primeiro *byte* das *words* de comunicação com o variador definidas no *harware* do projeto (figura 4.33), a função sabendo este *byte* vai ler e escrever nas *words* de controlo do variador corretas.

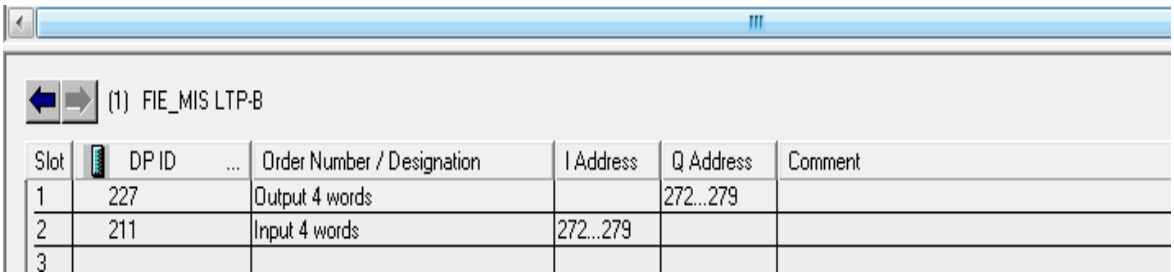


Figura 4.33: Identificação na parametrização do *hardware* das *words* de controlo e de estado.

Na criação da função para os variadores optou-se por utilizar FC em vez de FB porque não é necessário guardar valores ou estados de variáveis em memória, nem atribuir um DB de instância.

A *control word* antes de ser enviada, é separada em bits o que permite isolar o bit de *reset* do variador, o de paragem rápida e o de paragem com rampa. A *status word* quando é recebida também é desdobrada em bits e apenas os mais importantes são transpostos para saídas da FC, tais como o de erro, fins de curso e os de estado do variador.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

O controlo do sentido de rotação do motor é feito enviando velocidades positivas ou negativas para o variador.

Assim sendo, a FC ficou com a seguinte disposição de entradas e saídas.

- **Entradas**
  - *Enable* – Condição on/off da FC; (*Bool*)
  - *Direction_CW* – Rotação no sentido dos ponteiros do relógio; (*Bool*)
  - *Direction_CCW* – Rotação contrária ao sentido dos ponteiros do relógio; (*Bool*)
  - *Reset* – *Reset* dos erros; (*Bool*)
  - *Speed* – Velocidade pretendida (0-100% da velocidade nominal do motor); (*Real*)
  - *Ramp* – Rampa de aceleração/desaceleração em milissegundos, máximo de 32 segundos; (*Real*)

  Caso sejam necessárias rampas superiores, estas têm que ser parametrizadas diretamente no variador.

  - *Motor_Consumption* – Corrente nominal do motor; (*Real*)
  - *VEV_Max_Consumption* – Corrente máxima admitida pelo variador; (*Real*)

  Os dois últimos parâmetros de entrada servem para fazer um *scaling* à corrente atual recebida do LTP-B.

- *LADDR* – Número do byte inicial do conjunto de *words* utilizadas para a comunicação com o LTP-B; (*Int*)

➢ **Saídas**
- *Output_Enable* – Estágio de saída habilitada; (*Bool*)
- *Ready* – Conversor pronto a funcionar; (*Bool*)
- *PO_Enable* – Dados PO habilitados; (*Bool*)
- *Fault* – Irregularidade/aviso; (*Bool*)
- *Limit_CW* – Fim de curso sentido CW; (*Bool*)
- *Limit_CCW* – Fim de curso sentido CCW; (*Bool*)
- *Status_Code* – Estado do conversor; (*Byte*)
- *Error_Code* – Código da falha existente; (*Byte*)
- *Current_Speed* – Velocidade atual (0-100% da velocidade nominal do motor); (*Real*)
- *Current_Consumption* – Consumo atual (0-100% da corrente nominal do motor); (*Real*)

4.17. FC de Ligação ao Variador *MOVIMOT* da SEW

O funcionamento da FC criada para os *Movimots* (figura 4.34) é quase idêntica à criada para os LTP-B, mudando apenas uma das *words* recebidas. O *Movimot* em vez de enviar a velocidade atual do motor, envia uma segunda *status word* [9][10].

Figura 4.34: Variador *Movimot* acoplado a um motor.

A forma de comunicação entre o PLC e o *Movimot* também é um pouco diferente, enquanto no LTP-B a comunicação era direta, nos *Movimots* é necessário um módulo de interface chamado de MFP. O protocolo de comunicação entre o PLC e o MFP continua a ser o *profibus*. Estes módulos também possuem entradas e saídas digitais que podem ser utilizadas caso seja necessário.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

A forma como é feito o endereçamento indireto dentro da FC dos *Movimots* é igual à dos LTP-B.

Após terminada, a FC ficou com o seguinte aspeto em termos de entradas e saídas.

- **Entradas**
  - *Enable* – Condição on/off da FC; (*Bool*)
  - *Direction_CW* – Rotação no sentido dos ponteiros do relógio; (*Bool*)
  - *Direction_CCW* – Rotação contrária ao sentido dos ponteiros do relógio; (*Bool*)
  - *Reset* – *Reset* dos erros; (*Bool*)
  - *Speed* – Velocidade pretendida (0-100% da velocidade nominal do motor); (*Real*)
  - *Ramp_Ms* – Rampa de aceleração e desaceleração do motor; (*Real*)
  - *LADDR* – Número do byte inicial do conjunto de *words* utilizadas para a comunicação com o *Movimot*; (*Int*)

- **Saídas**
  - *Controller_Enable* – *Movimot* habilitado; (*Bool*)
  - *Unit_Enabled* – *Movimot* pronto a funcionar; (*Bool*)
  - *PO_Enable* – Dados PO habilitados; (*Bool*)
  - *Fault* – Irregularidade/aviso; (*Bool*)
  - *Status_Code* – Estado do conversor; (*Byte*)
  - *Error_Code* – Código da falha existente; (*Byte*)
  - *Current_Consumption* – Consumo atual (0-100% da corrente nominal do motor); (*Real*)
  - *Status_Word2* – Palavra de estado 2; (*Word*)

4.18. FC de Ligação ao Variador *MOVIDRIVE* da SEW

Este tipo de variador, ao contrário dos dois anteriores, permite controlo de posição, possui entradas e saídas virtuais e configuráveis e uma programação um pouco mais complicada, o que levou à criação de uma FC mais extensa do que as anteriores [11][12].

Em vez de 6 *words* de comunicação, este *driver* possui 8 ou mais, mas a FC criada apenas permite 8 *words* de comunicação (figura 4.35). Quatro delas são para controlo e as outras quatro são para estado. A comunicação pode ser por *profibus*, *ethernet*, entre outros protocolos, bastando para isso ter a carta necessária no variador.

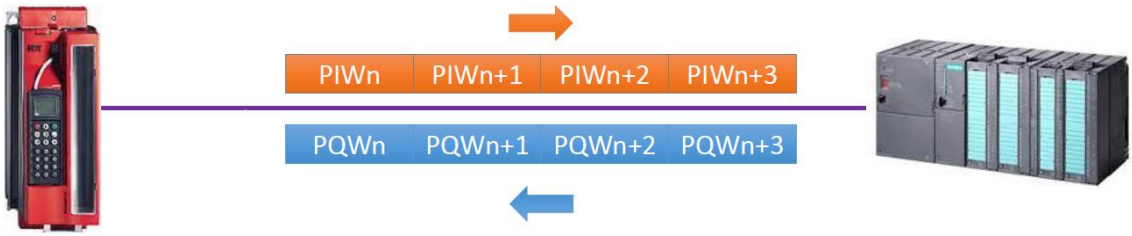


Figura 4.35: Ilustração das *words* de comunicação entre o *Movidrive* e um PLC *Siemens*.

As *words* de controlo enviadas para o variador são a *control word*, a posição pretendida em milímetros, a velocidade pretendida e a rampa. As que são recebidas pelo PLC são a, *status word*, posição atual, consumo atual e código de erro.

Da mesma forma que os dois variadores anteriores, a *status word* e a *control word* são desdobradas de forma a ser possível aceder aos bits individuais de ambas. A *status word* neste caso possui inseridos nela o estado dos terminais virtuais criados no variador.

Tanto a velocidade pretendida como a posição pretendida e atual necessitam que seja feito um *scaling* para que tanto o programa do PLC como o *driver* as possam interpretar corretamente, este *scaling* é feito no interior da FC.

Com estes variadores a escolha do sentido de rotação é feita internamente nele, dependendo da posição atual e da posição de destino pretendida.

Estes *drivers* para poderem fazer controlo de posição necessitam que o motor possua *encoder*.

O fluxograma de funcionamento desta função não pode ser mostrado neste relatório por falta de autorização da empresa.

A FC dos *Movidrive* ficou com as seguintes entradas e saídas.

- **Entradas**
  - *Enable* – Condição on/off da FC; (*Bool*)
  - *Stop* – Normal *stop*, paragem com rampa; (*Bool*)
  - *Rapid_Stop* – *Rapid stop* paragem com a rapa mínima definida nos parâmetros do variador; (*Boo*l)
  - *Reset* – *Reset* erro do variador; (*Bool*)
  - *LADDR* – Número do *byte* inicial do conjunto de *words* utilizadas para a comunicação com o *Movidrive*; (*Int*)
  - *Desired_Speed* – Velocidade pretendida; (*Real*)
  - *Desired_Position* – Posição pretendida; (*Real*)
  - *Ramp* – Tempo de aceleração e paragem do motor em milissegundos; (*Real*)
  - *DI10_Reference* – Referenciar posição; (*Bool*)
  - *DI11_Positioning_Aut* – Posicionar automático; (*Bool*)
  - *DI12_Positioning_Man* – Posicionar manual (*Bool*)
  - *DI13_JOG_CW* – Movimento de JOG no sentido dos ponteiros do relógio; (*Bool*)
  - *DI14_JOG_CCW* – Movimente de JOG no sentido contrário aos ponteiros do relógio; (*Bool*)
  - DI15 – Entrada digital sem utilização; (*Bool*)
  - DI16 – Entrada digital sem utilização; (*Bool*)
  - DI17 – Entrada digital sem utilização; (*Bool*)

- **Saídas**
  - *Output_Enabled* – Saídas habilitadas; (*Bool*)
  - *Ready – Movidrive* pronto a funcionar; (*Bool*)
  - *PO_Enable* – Dados PO habilitados; (*Bool*)
  - *Fault* – Irregularidade/aviso; (*Bool*)
  - *Limit_CW* – Fim de curso sentido CW; (*Bool*)
  - *Limit_CCW* – Fim de curso sentido CCW; (*Bool*)
  - *DO10_In_Position* – Em posição; (*Bool*)

- *DO11_Referenced* – Referenciado; (*Bool*)
- DO12 – Saída digital; (*Bool*)
- DO13 – Saída digital; (*Bool*)
- DO14 – Saída digital; (*Bool*)
- DO15 – Saída digital; (*Bool*)
- DO16 – Saída digital; (*Bool*)
- DO17 – Saída digital; (*Bool*)
- *Current_Position* – Posição atual; (*Real*)
- *Current_Consumption* – Consumo atual (0-100% da corrente nominal do motor); (*Real*)
- *Error_Code* – Código da falha existente; (*Byte*)

Com esta função fica terminada a exposição de todas as funções de controlo de maquinaria parar a indústria cerâmica desenvolvidas durante o estágio.

Durante o decorrer do estágio surgiram situações em que se revelou vantajoso desenvolver outras capacidades para além das referidas neste capítulo, esses temas não são abordados porque estariam um pouco fora do âmbito do estágio.

5. TESTES, RESULTADOS E AVALIAÇÃO

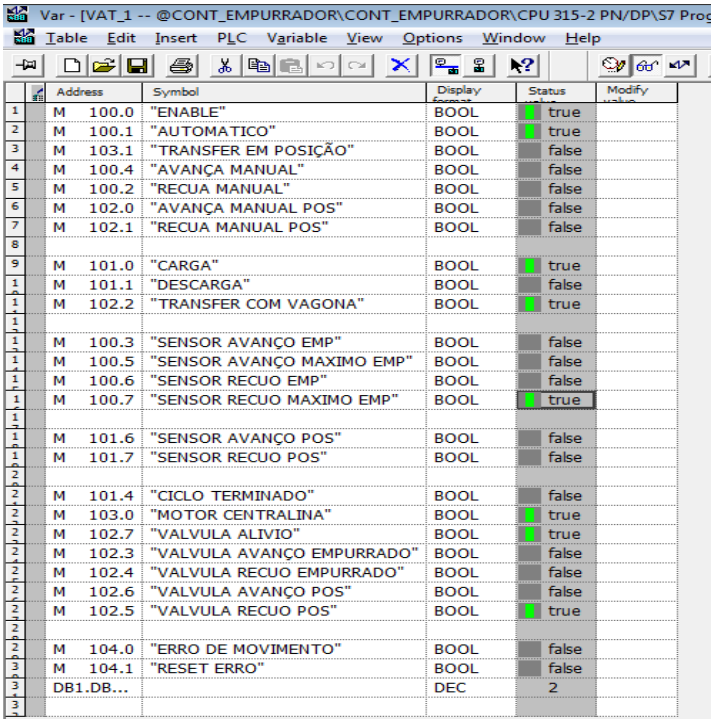
5.1. Testes

Durante o desenvolvimento das funções, e após terminadas foi necessário proceder ao seu teste. Os testes feitos tinham como objetivo observar se as funções respondiam da forma esperada ao que lhes era pedido, se ativavam as saídas corretas tendo em conta as entradas ligadas.

O *software* de programação de autómatos da *Siemens* (*Simatic Manager*) possui uma ferramenta (PLC Sim) que permite simular um autómato no computador, modificar os estados das entradas e observar como as saídas do PLC se comportam. Esta ferramenta revelou-se muito útil durante a fase de *debuging* das funções. Além desta ferramenta foram disponibilizados autómatos da *Siemens* da gama 315 para testes.

Para as funções que controlam motores com auxílio de variadores, foram usados variadores e motores de teste existentes na empresa.

Nos testes realizados às funções revelou-se necessária a observação da mudança de estado ou de valor de determinadas variáveis (figura 5.1). Esta observação pôde ser feita recorrendo a tabelas de variáveis dentro do *Simatic Manager*, que permitem a visualização em *runtime* de variáveis selecionadas pelo operador.



	Address	Symbol	Display format	Status value	Modify value
1	M 100.0	"ENABLE"	BOOL	true	
2	M 100.1	"AUTOMATICO"	BOOL	true	
3	M 103.1	"TRANSFER EM POSIÇÃO"	BOOL	false	
4	M 100.4	"AVANÇA MANUAL"	BOOL	false	
5	M 100.2	"RECUA MANUAL"	BOOL	false	
6	M 102.0	"AVANÇA MANUAL POS"	BOOL	false	
7	M 102.1	"RECUA MANUAL POS"	BOOL	false	
8					
9	M 101.0	"CARGA"	BOOL	true	
1	M 101.1	"DESCARGA"	BOOL	false	
1	M 102.2	"TRANSFER COM VAGONA"	BOOL	true	
1					
1	M 100.3	"SENSOR AVANÇO EMP"	BOOL	false	
1	M 100.5	"SENSOR AVANÇO MAXIMO EMP"	BOOL	false	
1	M 100.6	"SENSOR RECUO EMP"	BOOL	false	
1	M 100.7	"SENSOR RECUO MAXIMO EMP"	BOOL	true	
1					
1	M 101.6	"SENSOR AVANÇO POS"	BOOL	false	
1	M 101.7	"SENSOR RECUO POS"	BOOL	false	
2					
2	M 101.4	"CICLO TERMINADO"	BOOL	false	
2	M 103.0	"MOTOR CENTRALINA"	BOOL	true	
2	M 102.7	"VALVULA ALIVIO"	BOOL	true	
2	M 102.3	"VALVULA AVANÇO EMPURRADO"	BOOL	false	
2	M 102.4	"VALVULA RECUO EMPURRADO"	BOOL	false	
2	M 102.6	"VALVULA AVANÇO POS"	BOOL	false	
2	M 102.5	"VALVULA RECUO POS"	BOOL	true	
2					
2	M 104.0	"ERRO DE MOVIMENTO"	BOOL	false	
3	M 104.1	"RESET ERRO"	BOOL	false	
3	DB1.DB...		DEC	2	
3					

Figura 5.1: Estado das variáveis de simulação de uma função.

Em relação aos PID's, e de forma a se conseguir testá-los, utilizou-se uma função das bibliotecas da *Siemens* para esse efeito. Esta função quando interligada com o PID altera os valores do *presente value* em função da saída do controlador.

A conjugação de todas estas ferramentas de teste revelou-se bastante importante no desenvolvimento dos FB e FC pois permitiram a observação de erros existentes e consequente resolução.

Apesar de se ter conseguido testar as funções em ambiente simulado e de laboratório, o verdadeiro teste destas apenas será feito quando colocadas a trabalhar inseridas num programa completo de uma fábrica. No final do estágio começou-se a inserir as funções no projeto de uma empresa cerâmica, este projeto só será implementado alguns meses depois do final do estágio.

Outro dos testes efetuados às funções durante o estágio foi, na eventualidade de haver uma falha de energia, se o PLC guarda o estado das variáveis internas das funções. Como os FB possuem DB de instância onde são guardadas todas as variáveis da FB exceto as temporárias e uma vez que estes DB são retentivas, sempre que se desliga o PLC é guardado o estado das variáveis dos DB. Esta característica foi possível observar nos testes feitos às funções durante o estágio.

5.2. Resultados

Após testadas, todas as funções revelaram estar a funcionar dentro dos parâmetros estipulados para a sua correta execução, mesmo não se podendo afirmar que estejam prontas a trabalhar sem ser necessário nenhum ajuste, pois apesar de terem sido exaustivamente testadas, não se tornou possível recriar exatamente as situações de funcionamento que a máquina para a qual se destinam sofre. Durante os testes realizados e simulações ao funcionamento da funções, todas as especificações de funcionamento teórico necessárias cumprir foram correspondidas.

Como a segurança no trabalho é um dos requisitos mais importante na hora de programar um controlador de uma máquina, todas as situações passiveis de provocar acidentes ou de danificar a máquina foram tidas em conta e as funções projetadas de forma a, ser possível, bloquear o funcionamento ou gerar o alarme necessário ao bloqueio da máquina.

Outra das situações que foi necessário prestar atenção, foi em algumas das funções respeitar as sequências de arranque e paragem da máquina de forma a não danificar nenhum componente desnecessariamente. Esta situação assim como as referidas antes também foram cumpridas e testadas.

5.3. Avaliação

A implementação destas funções em programas de controlo facilita a programação dos PLC. As funções foram criadas de forma a serem *standard* e possíveis de aplicar a máquinas com especificações diferentes, por exemplo a função dos queimadores pulsados pode controlar queimadores com um número de bicos desde um até 32, este atributo revela-se uma mais-valia para a implementação da programação em máquinas para a indústria cerâmica.

Tendo em conta que a tecnologia se encontra em constante evolução, daqui a alguns anos as funções criadas no âmbito do estágio podem se revelar obsoletas e ultrapassadas mas neste momento revelam-se uma mais-valia para a programação das linhas de produção na indústria

cerâmica acelerando a criação do programa e facilitando a colocação desse mesmo programa em funcionamento.

6. CONCLUSÃO

Quanto mais rápido, fiável e duradouro se revelar um programa de automação mais interesse este suscitará junto de possíveis clientes. Foi a pensar nestas qualidades que se criaram as funções descritas neste relatório, funções estas que além de diminuírem a complexidade do código final do PLC, tornam o programa fácil de colocar em funcionamento e de compreender. Estas vantagens aliadas à abrangência de combinações de funcionamento que as funções permitem, tornaram a sua criação bastante interessante e desafiante.

Apesar do conhecimento acerca do funcionamento das diferentes etapas de produção de uma indústria cerâmica ser bastante limitado no início do estágio o que dificultou a princípio compreender o correto funcionamento das máquinas e por consequente das funções, foi bastante empolgante apreender e desenvolver conteúdos de forma a melhorar os processos já existentes.

Como descrito no capítulo anterior, durante a criação das funções foram realizados testes de forma a garantir que estas funcionavam como esperado. Um dos pontos que não revelou provocar problemas durante os testes foi a utilização de memória temporária para algumas variáveis. Este tipo de memória é utilizada por todo o programa a cada ciclo do PLC, não ficando limitada às variáveis criadas como temporárias. Ao contrário da memória estática que retém o estado das variáveis entre ciclos, a temporária não o faz, o que se revelou um erro de conceção das funções quando se colocaram em funcionamento inseridas num programa mais extenso e que necessitasse de mais memória para o seu funcionamento. Esta situação fez com que algumas funções não respondessem como o esperado. Durante os testes das funções este erro não foi detetado porque os programas feitos para os testes apenas continham as funções em si, o que necessita de muito pouca memória, e a memória temporária não era reescrita entre ciclos do PLC. Mas rapidamente se encontrou uma solução para o problema, bastando para isso passar as variáveis mais sensíveis e passiveis de alteração do seu estado para variáveis do tipo estático o que resolveu o problema. De salientar que inicialmente se colocaram algumas das variáveis como sendo temporárias de forma a diminuir a memória do PLC necessária para a função.

Mesmo tendo em conta a situação descrita anteriormente e a necessidade de ajustar alguns tempos de funcionamento das funções, conclui-se que o seu funcionamento e aplicação foram conforme o esperado e que estas contribuem para o melhor desempenho tanto das máquinas como do próprio autómato.

A opção de realização de estágio para conclusão do mestrado revelou-se como sendo a melhor opção possível de tomar, pois permitiu conhecer um pouco mais do mundo do trabalho, adquirir conhecimentos ou aprofundar os já existentes, e estar em contacto com tecnologias e formas de fazer novas. Também permitiu ficar a conhecer um tipo de indústria da qual não tinha grandes conhecimentos e de me possibilitar estar presente num projeto de implementação de uma fábrica nova.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] SITRAIN (ST-PRO1). SIMATIC S7 Programação 3 (ST-PRO1 Course), Version 5.3.

[2] PROFIBUS Network Manual. SIMATIC NET System Manual, Edition 04 / 2009. SIEMENS.

[3] SITRAIN (ST-PRO2). SIMATIC S7 Programação 3 (ST-PRO2 Course), Version 5.3.

[4] SITRAIN (ST-PRO3). SIMATIC S7 Programação 3 (ST-PRO3 Course), Version 5.3.

[5] Configuring Hardware and Communication Conections with STEP 7. SIMATIC Manual, Edition 03/2006. SIEMENS.

[6] João R. Caldas Pinto (2010). *Técnicas de Automação, 3ª Edição*. Edição em Português, ETEP - Edições Técnicas e Profissionais Março de 2010.

[7] Ladder Logic (LAD) for S7-300 and S7-400 Programing, SIMATIC Reference Manual, Edition 05/2010. SIEMENS.

[8] MOVITRAC LTP-B. Operating Instructions, Edition 03/2013 20091745 / EN. SEW-EURODRIVE.

[9] MOVIMOT MM03B to MM30B. Operating Instructions, Edition 10/2000 10505814 / EN. SEW-EURODRIVE.

[10] MOVIMOT MM..D com motor trifásico DT/DV. Instruções de Operação, Edição 08/2009 16817249 / PT. SEW-EURODRIVE.

[11] MOVIDRIVE MDX60B / 61B. Instruções de Operação, Edição 01/2010 16837649 / PT. SEW- EURODRIVE.

[12] MOVIDRIVE MDX61B. DriveSync via Fieldbus Application, Edição 08/2010 17004411 / EN. SEW-EURODRIVE.

[13] Página web Suporte Siemens (02/2014). http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo2&aktprim=99&lang=en

[14] Página web SEW-EURODRIVE (03/2014). http://www.seweurodrive.com/adressen/country.php?land=Portugal&kontinent=europe

[15] S7-300 CPU31xC and CPU 31x: Technical specifications. SIMATIC Manual, Edition 03 / 2011. SIEMENS.

[16] Statement List (STL) for S7-300 and S7-400 Programming. SIMATIC S7 Reference manual, Edition 10 / 98 Release 01. SIEMENS.

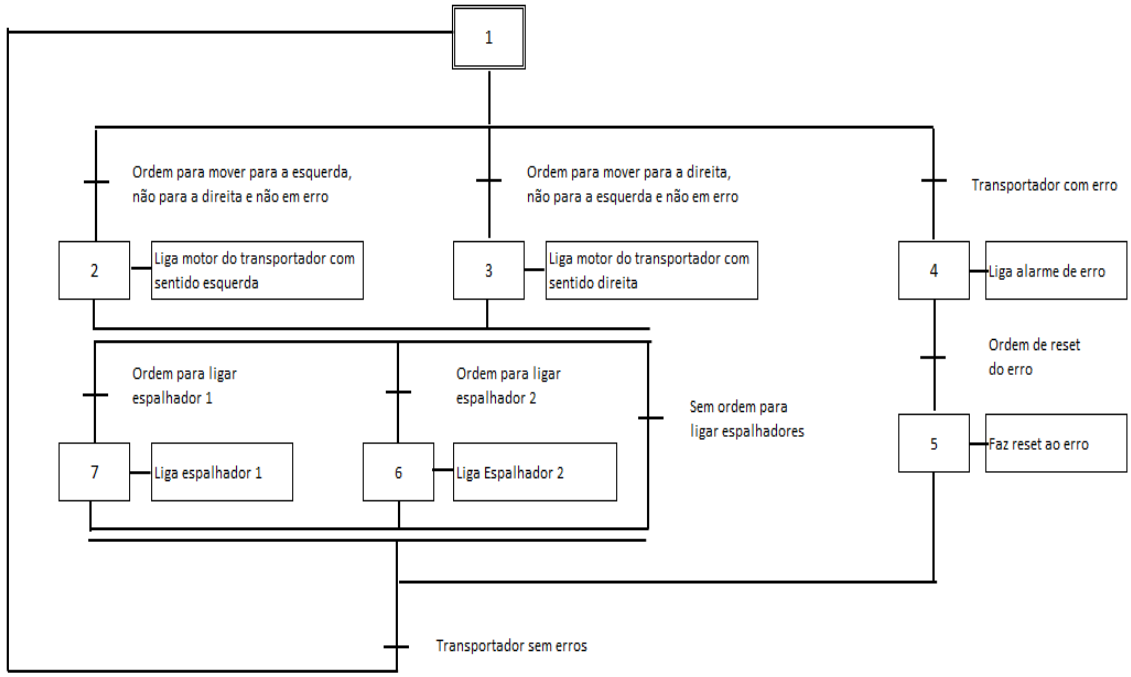
[17] STEP 7 Professional V12.0. SIMATIC System Manual, Edition 01 / 2013. SIEMENS.

Anexo A

Grafcet de nível 1

Este anexo pretende mostrar o *grafcet* de nível 1 da função das telas de transporte criada no decorrer do estágio.

Telas de Transporte



Anexo B

Grafcet de nível 1

Este anexo pretende mostrar o *grafcet* de nível 1 da função do empurrador do transfer criada no decorrer do estágio.

Controlo do Empurrador do Transfer