

Desenvolvimento de uma aplicação para geração automática de código para o PLC de controlo de um *shuttle car*

Henrique Miguel Afonso Domingos

Dissertação de Mestrado

Orientador na FEUP: Professor Doutor Germano Correia dos Santos Veiga

Orientador na Consoveyo, S.A.: Engenheiro Vitor Vaz



Mestrado Integrado em Engenharia Mecânica

Junho 2018

“Tudo vale a pena se a alma não é pequena.”

Fernando Pessoa

Resumo

Apesar da crescente complexidade dos sistemas automáticos e da constante necessidade em customizar soluções, a pressão sobre os custos e tempos de execução mantém uma tendência crescente. Neste sentido, os fabricantes têm procurado alternativas para agilizar a programação deste tipo de sistemas. A geração de código representa uma destas alternativas.

A geração de código para Controladores Lógicos Programáveis (PLCs) é uma realidade complexa e inovadora, não havendo, até ao momento, relevante implementação ao nível industrial. No entanto, a comunidade científica tem apontado grandes vantagens na utilização desta técnica.

A presente dissertação pretende contribuir para a investigação desta realidade, através do desenvolvimento de uma aplicação de software para a geração automática de código para o PLC de controlo de um *shuttle car*. Esta dissertação foi desenvolvida em colaboração com o departamento de Automação da Consoveyo, S.A..

São exploradas diferentes ferramentas como o TIA Portal Openness® e a biblioteca de comunicação Sharp 7, estabelecendo-se uma comparação entre a geração de código e a parametrização de código a partir da comunicação com o PLC. Com base nas suas capacidades e nos princípios de funcionamento dos armazéns concebidos pela Consoveyo, S.A., é estabelecida uma abordagem com vista ao desenvolvimento da aplicação de software.

De modo a atender aos requisitos necessários à geração de código, foi desenvolvido um programa modular, no TIA Portal®, para o controlo do *shuttle car*, o qual foi testado no simulador FactoryIO®.

Usando o TIA Portal Openness® e métodos de manipulação de documentos XML, obteve-se uma aplicação de geração de código que possibilita a criação do programa de controlo do *shuttle car*, de acordo com a configuração definida para o sistema.

Palavras-chave: programação de PLC, geração de código para PLC, armazém automático, fabrico integrado por computador.

Development of a software application for automatic generation of PLC code to control a shuttle car

Abstract

Despite the increasing complexity of automated systems and the constant need for delivering customized solutions, the pressure on costs and execution times continues to grow. Thus, several manufacturers are looking for solutions, such as code generation, to speed up the programming task.

The code generation for Programmable Logic Controllers (PLCs) is a complex and innovative concept, and, so far, there is no significant implementation in the industry. However, the scientific community has pointed out great advantages on the usage of this technique.

This thesis aims to contribute to this area of research by developing a software application to generate the PLC code that controls a shuttle car. It was developed in collaboration with Consoveyo, S.A. automation department.

Several tools, such as TIA Portal Openness® and Sharp7 communication library are explored in depth, establishing a comparison between code generation and code parameterization through communication with the PLC. Based on its capabilities and on the operating principles of the systems designed by Consoveyo, S.A, an approach is established for the development of the software application.

In order to meet the requirements needed for generating PLC code, a modular control program for the shuttle car was developed in TIA Portal® and tested using FactoryIO® simulator.

Using TIA Portal Openness® and XML manipulation methods a application for automatic code generation was developed. This application creates the program needed to control the shuttle car according to the defined system configuration.

Keywords: PLC programming, PLC code generation, automated warehouse, computer integrated manufacturing.

Agradecimentos

Gostaria de expressar os meus mais sinceros agradecimentos a todos os que, direta ou indiretamente contribuíram para o meu sucesso ao longo destes cinco anos, em particular:

Ao Professor Doutor Germano Manuel Correia dos Santos Veiga, por toda a dedicação, partilha de conhecimentos e acompanhamento prestado ao longo da presente dissertação.

Ao Engenheiro Vitor Vaz, pela disponibilidade e pela oportunidade de realizar este projeto em colaboração com a Consoveyo, S.A.

Sem a contribuição de ambos a concretização deste projeto não teria sido possível.

À Consoveyo, S.A. pela oportunidade de realizar este projeto, que em muito contribuiu para a minha formação pessoal e profissional. Contactar com a realidade laboral e industrial foi uma experiência enriquecedora, que contribuirá, certamente, para o meu futuro enquanto engenheiro.

À Sra. Sara Fernandes, pela disponibilidade que demonstrou desde o primeiro contacto efetuado e por todo o apoio prestado ao longo do processo de escolha da dissertação.

Ao Professor Doutor Fernando Gomes de Almeida pelos conselhos sábios e pela calma transmitida num momento menos positivo desta dissertação.

Aos meus pais, Maria da Conceição Domingos e Henrique Domingos, por serem uma figura de exemplo e por todo o apoio anímico e confiança que me transmitiram ao longo de todo o meu percurso académico, e em especial durante esta dissertação.

À minha irmã, Sofia Domingos, por toda ajuda, disponibilidade e críticas construtivas dadas ao longo deste trabalho.

À minha namorada, Ana Lúcia Poças pela paciência, companhia, amizade, apoio, por nunca me deixar desistir e por constantemente me desafiar a dar o meu melhor.

Aos colaboradores da Consoveyo, S.A., em especial ao Engenheiros Pedro Carneiro e Édgar Magalhães, pela disponibilidade e críticas construtivas dadas ao longo de todo este trabalho. Também pelo companheirismo e pelos momentos de pausa sempre divertidos.

Aos meus colegas do curso de Engenharia Mecânica e aos meus amigos de Viseu por todos os bons momentos que partilhámos ao longo destes cinco anos.

A todos, os meus mais sinceros agradecimentos ...

Índice de Conteúdos

1	Introdução	1
1.1	Enquadramento da dissertação e motivação	1
1.2	Desenvolvimento de uma aplicação para geração automática de código para o PLC de controlo de um <i>shuttle car</i> na Consoveyo, S.A	1
1.3	Objetivos da dissertação	2
1.4	Método seguido na dissertação	2
1.5	Estrutura da dissertação	4
2	Estado da Arte	5
2.1	Armazéns automáticos	5
2.2	Controladores Lógicos Programáveis	5
2.2.1	História dos PLCs	6
2.2.2	Norma IEC 61131	6
2.3	Práticas aplicadas na Consoveyo, S.A.	8
2.3.1	Linguagens de programação	8
2.3.2	Plataformas utilizadas: TIA Portal® e SIMATIC STEP 7®	8
2.4	Flexibilização da programação de PLCs	9
2.4.1	Geração automática de código	10
2.4.2	Parametrização de código a partir da comunicação com o PLC	14
3	Sistema, problema em causa e ferramentas exploradas	15
3.1	Descrição geral do sistema	15
3.2	Princípio de funcionamento do <i>shuttle car</i>	18
3.2.1	Constituintes e controlo do <i>shuttle car</i>	18
3.2.2	Modos de Operação	19
3.2.3	Variantes possíveis	20
3.3	Definição do problema e abordagens	20
3.4	Ferramentas exploradas	21
3.4.1	TIA Portal Openness	21
3.4.2	Sharp 7	30
3.5	Abordagem escolhida	33
4	Programa de controlo e ambiente de simulação desenvolvidos	35
4.1	Ambiente de simulação	35
4.1.1	Sistema a controlar	35
4.1.2	Funcionamento do sistema controlado	37
4.2	Programa de Controlo do <i>shuttle car</i>	38
4.2.1	Requisitos de Modularidade	38
4.2.2	Componentes do programa desenvolvido em TIA Portal®	39
5	Aplicação desenvolvida	43
5.1	Objetivo da aplicação	43
5.2	Arquitetura da aplicação	44
5.3	Interface da aplicação	45
5.4	Funcionamento da aplicação	45
5.4.1	Adição de uma nova linha	46
5.4.2	Gravação dos ficheiros XML	47
5.4.3	Importação dos ficheiros XML para o projeto modelo	47
5.5	Geração de código a partir da manipulação de documentos XML	49
5.5.1	Estrutura dos documentos XML do TIA Portal®	49
5.5.2	Documentos XML modelo	52
5.5.3	Método de Manipulação dos ficheiros XML do TIA Portal®	53
5.5.4	Manipulações efetuadas nos Documentos XML do TIA Portal®	53
6	Conclusões e perspetivas de trabalho futuro	59
6.1	Conclusões	59
6.2	Trabalhos futuros	59
	Referências	61
ANEXO A:	Aplicação desenvolvida	63
ANEXO B:	Programa desenvolvido no TIA Portal	104

Siglas

AGV – Automated Guided Vehicle

API – Application Programming Interface

CAD – Computer-Aided Design

CAM – Computer-Aided Manufacturing

CNC – Computer Numeric Control

DB – Database Block

DLL – Dinamic-Link Library

FB – Function Block

FBD – Function Block Diagram

HMI – Human-Machine Interface

IDE – Integrated Development Environment

IEC – International Eletrotechnical Comission

IL – Instruction List

LD – Ladder Diagrams

OB – Organization Block

PLC – Programmable Logic Controller

RGV – Railed Guided Vehicle

SFC - Sequential Function Chart

ST – Structured Text

W3C – World Wide Web Consortium

XML – Extensible Markup Language

Índice de Figuras

Figura 1 – Diagrama de Gantt: Tarefas realizadas ao longo da dissertação.....	3
Figura 2 – Exemplo de <i>shuttle car</i> produzido pela Consoveyo.....	5
Figura 3 – Arquitetura do software descrita pela norma IEC 61131-3.	7
Figura 4 – <i>Design</i> do processo para sistemas de controlo lógicos..	12
Figura 5 – <i>Layout</i> geral de um armazém automático.	15
Figura 6 – Zonas funcionais 1,2 e 3	16
Figura 7 – Zonas funcionais 4 e 5	17
Figura 8 – Zona funcional 11	17
Figura 9 – Zona funcional 12	17
Figura 10 – Contituíntes de um <i>shuttle car</i> produzido pela Consoveyo..	18
Figura 11 – Painel de controlo do carro de transferência	19
Figura 12 – Public API.	21
Figura 13 – Configurações do TIA Portal.	22
Figura 14 – Interface gráfica da aplicação <i>StartOpenness</i> ;	23
Figura 15 – Estrutura dos ficheiros XML.....	25
Figura 16 – Prólogo do ficheiro XML.....	25
Figura 17 – Comentários nos ficheiros XML.....	26
Figura 18 – Exemplo de XML Schema.	26
Figura 19 – Implementação do XmlTextReader e XmlTextWriter.....	27
Figura 20 – Implementação do XmlDocument.	27
Figura 21 – Implementação do XmlSerialize a partir das classes manuais.....	28
Figura 22 – Obtenção das classes.	28
Figura 23 - Classes obtidas.....	29
Figura 24 – Implementação do XmlSerialize a partir das classes automáticas.	29
Figura 25 – Programa em TIA Portal para teste do Sharp 7.....	31
Figura 26 – Criação do servidor para conexão ao PLC a partir do NetToPLCsim.	31
Figura 27 – Implementação da função ConnectTo().	31
Figura 28 – Implementação da função DBRead().	32
Figura 29 – Implementação da função s7Multivar_.Write().	32
Figura 30 – Função de captura dos valores atuais.	33
Figura 31 – Transferência dos valores atuais para os valores iniciais.	33
Figura 32 – <i>Layout</i> desenvolvido no FactoryIO.....	36
Figura 33 - <i>Layout</i> desenvolvido no FactoryIO (Vista superior).	36
Figura 34 – Esquematização do <i>layout</i> desenvolvido.	36
Figura 35 – Painel de controlo do quadro elétrico.....	37
Figura 36 – Esquematização da modularização do sistema em estudo.	38

Figura 37 – Indexação de um vetor através de uma variável inteira.	39
Figura 38 – GRAFCET do <i>function block</i> Conv_1	40
Figura 39 - GRAFCET do <i>function block</i> Shuttle_Load_Unload.	41
Figura 40 – Lista de ordens de transporte a cumprir pelo <i>shuttle car</i>	42
Figura 41 – Sistemas com <i>layouts</i> distintos.	43
Figura 42 – Sistemas com <i>layouts</i> distintos em ambiente de simulação.	43
Figura 43 – Arquitetura da aplicação.	44
Figura 44 – Projeto modelo no TIA Portal.	44
Figura 45 – Interface gráfica da aplicação.....	45
Figura 46 – Esquematização das manipulações necessárias na adição de uma nova linha de descarga.	46
Figura 47 – Gravação dos documentos XML manipulados.	47
Figura 48 – Iniciação de uma nova instância do TIA Portal.	48
Figura 49 – Abertura do projeto modelo.	48
Figura 50 – Acesso e importação para a área <i>Program Blocks</i>	49
Figura 51 – Acesso e importação para a área <i>PLC Tags</i>	49
Figura 52 – Estrutura geral dos ficheiros XML do TIA Portal.	50
Figura 53 – Designações da área <i>SW.CodeBlock</i> para os componentes.....	50
Figura 54 – Interface de um componente.	50
Figura 55 – Estrutura do XML de um <i>organization block</i>	51
Figura 56 – Exemplo geral da relação entre <i>Parts</i> e <i>Wires</i>	52
Figura 57 – Documento XML modelo para o FB Order_Sensor.	52
Figura 58 – Criação das classes necessárias à manipulação dos ficheiros XML via XmlSerialize.	53
Figura 59 – Elementos de software do projeto desenvolvido manipulados pela aplicação.	54
Figura 60 – Instanciação dos FBs responsáveis pelo controlo dos <i>conveyors</i> no OB.....	55
Figura 61 – Interface do FB Dest_Sensor (sistema A).....	55
Figura 62 – Variáveis/comandos usados na <i>network</i> (sistema A).....	56
Figura 63 – Interface do FB Dest_Sensor (sistema B).....	56
Figura 64 – Variáveis/comandos usados na <i>network</i> (sistema B).....	56
Figura 65 – <i>Tagtable</i> Inputs (sistema A).....	57
Figura 66 – <i>Tagtable</i> Inputs (sistema B).....	58

Índice de Tabelas

Tabela 1 – Exemplos de funcionalidades e respetivos métodos implementáveis através do TIA Portal Openness®	24
Tabela 2 – Funções disponíveis no Sharp7	30

1 Introdução

A crescente exigência das indústrias e o ambiente comercial fortemente competitivo vivido na atualidade tem criado novos desafios para os fornecedores de soluções automáticas. Fruto desta realidade, a pressão sobre os custos e tempos de execução deste tipo de soluções tem vindo a aumentar, o que tem levado à procura de alternativas. A geração de código tem sido apontada como uma possível solução para este problema, visando flexibilizar e agilizar a tarefa da programação dos sistemas automáticos.

É neste contexto que surge o tema da presente dissertação: Desenvolvimento de uma aplicação para a geração automática de código para o Controlador Lógico Programável (*Programmable Logic Controller* – PLC) de um *shuttle car*. Esta dissertação foi realizada no âmbito da unidade curricular de Dissertação do curso de Mestrado Integrado em Engenharia Mecânica, ramo de Automação, da Faculdade de Engenharia da Universidade do Porto e desenvolvida em colaboração com a Consoveyo, S.A.. Foi realizada durante o segundo semestre do ano letivo 2017/2018 e contou com a orientação do Professor Doutor Germano Manuel Correia dos Santos Veiga e do Engenheiro Vitor Vaz da Consoveyo, S.A..

1.1 Enquadramento da dissertação e motivação

A presente dissertação centra-se no desenvolvimento de uma aplicação que gere, de forma automática, o código do programa de controlo do PLC de um *shuttle car*, e foi elaborada em colaboração com o departamento de Automação da Consoveyo, S.A..

Os *shuttle cars* são equipamentos vastamente implementados em sistemas logísticos e permitem transportar cargas entre diferentes localizações. Caracterizados por uma elevada flexibilidade e rapidez, são normalmente aplicados em armazéns automáticos para a alimentação e recolha das paletes, bem como para a alimentação de estações de separação de pedidos.

Os armazéns automáticos da Consoveyo, S.A. incluem uma grande diversidade de equipamentos, entre os quais os *shuttle cars*, e são concebidos de acordo com os requisitos dos seus clientes, o que resulta em múltiplas configurações destes sistemas. Assim, o objetivo da empresa passa por flexibilizar o desenvolvimento dos programas de PLC que os controlam, de modo a que estes possam ser automaticamente adaptados às variações requeridas.

1.2 Desenvolvimento de uma aplicação para geração automática de código para o PLC de controlo de um *shuttle car* na Consoveyo, S.A

A Consoveyo, S.A. é uma empresa líder mundial no desenvolvimento de sistemas automáticos de transporte e armazenamento de materiais (Consoveyo 2017b). Anteriormente conhecida como Efacec *Handling Solutions*, foi adquirida pelo grupo alemão Körber em 2015 como parte do processo de internacionalização do grupo na área de negócio dos sistemas de logística (Consoveyo 2017a). Para além dos serviços nesta nova área, o grupo oferece aos seus clientes soluções nas áreas de automação, máquinas-ferramenta, sistemas farmacêuticos, papel, tabaco e investimentos em capitais de risco (Consoveyo 2017b).

A Consoveyo, S.A. concebe e constrói soluções automáticas “chave na mão”, personalizadas de acordo com os requisitos do cliente, de entre as quais: armazenamento, manuseamento, transporte interno, preparação de encomendas e software. Estas soluções são totalmente desenvolvidas no seio da empresa, desde o seu planeamento à integração no cliente, passando pelo *design*, montagem e desenvolvimento de software (Consoveyo 2017b). O portefólio da empresa dispõe de vários produtos, tais como: transelevadores, veículos guiados automaticamente (*Automated Guided Vehicles* – AGVs), veículos guiados sobre carris (*Rail*

Guided Vehicles – RGVs), carros de transferência (*shuttle cars*) e transportadores (*conveyors*). Além de atuar como fornecedor de soluções e produtos, a Consoveyo, S.A. dispõe ainda de uma vasta gama de serviços.

O departamento de Automação da Consoveyo, S.A. é responsável pelo projeto elétrico dos equipamentos e pelo desenvolvimento do software de controlo de baixo nível dos sistemas concebidos. Neste momento, um dos objetivos do departamento passa por encontrar alternativas para agilizar e flexibilizar a programação dos vários equipamentos concebidos. Isto porque, uma vez que a Consoveyo, S.A. produz soluções ajustadas aos requisitos de cada cliente, cada projeto desenvolvido resulta na implementação de sistemas com diferentes *layouts* e equipamentos com princípios de funcionamento distintos. Desta forma, a cada novo projeto é necessário modificar manualmente os programas que controlam o sistema implementado. Esta tarefa é bastante morosa e suscetível a erros, para além de necessitar de mão de obra qualificada na área da Automação. Uma ferramenta de geração de código permitiria agilizar e flexibilizar a programação dos equipamentos, contribuindo para menores tempos de execução dos projetos e, consequentemente, maior produtividade, o que traria vantagens ao nível competitivo para a empresa. Neste sentido, foi sugerido pela empresa desenvolver uma aplicação de software que permitisse gerar o programa de controlo dos *shuttle cars*, perante uma certa gama de variações no seu *layout* e princípio de funcionamento.

1.3 Objetivos da dissertação

Como referido anteriormente, o objetivo da presente dissertação passa pelo desenvolvimento de uma aplicação de software para a geração automática de código para o PLC de controlo de um *shuttle car* da Consoveyo, S.A. De modo a atingir o referido objetivo foi necessário:

- Averiguar as possibilidades existentes no âmbito da geração de código;
- Desenvolver código modular e eficiente para PLC, de modo a poder ser utilizado por uma ferramenta de geração de código;
- Desenvolver uma aplicação de software para geração automática de código e respetiva interface gráfica a partir da qual se pudesse definir a configuração desejada para o *shuttle car*.

1.4 Método seguido na dissertação

Dado o carácter inovador da temática da dissertação, havia alguma incerteza acerca da possibilidade de concretizar o último objetivo apresentado na secção anterior. Além disso, o desenvolvimento de uma ferramenta de geração de código na Consoveyo, S.A enfrentava diversas restrições, uma vez que teria de ser compatível com as ferramentas e procedimentos utilizados na empresa.

Neste sentido, optou-se por generalizar o tema da dissertação de forma a torná-lo mais abrangente, tendo-se optado por explorar as possibilidades existentes no que diz respeito à flexibilização da programação de PLCs.

Assim sendo, no início da presente dissertação efetuou-se pesquisa bibliográfica, subdividindo-se o tema em três assuntos principais, nomeadamente: armazéns automáticos, PLCs e flexibilização da programação de PLCs. A informação recolhida facilitou a contextualização do tema e permitiu identificar duas possíveis abordagens para a flexibilização da programação de PLCs: a geração de código e a parametrização de código a partir da comunicação com o PLC. Além disso, a pesquisa revelou a existência de diversas ferramentas, úteis para cada uma das referidas abordagens.

Posteriormente, seleccionou-se uma ferramenta para cada abordagem, com base na análise da sua viabilidade e das respetivas potencialidades. Para além disso, foi considerada a sua compatibilidade com as ferramentas e procedimentos utilizados na Consoveyo, S.A. Assim, optou-se pelo TIA Portal Openness® e pelo Sharp 7, respetivamente, para a geração de código e para a parametrização de variáveis através da comunicação com o PLC.

A continuidade da pesquisa revelou que a abordagem da geração de código seria de facto mais vantajosa, uma vez que permitiria uma maior flexibilidade na programação, bem como a obtenção de um programa de controlo menos pesado e mais fácil de manter. Deste modo, optou-se por utilizar o TIA Portal Openness®.

O recurso ao TIA Portal Openness® implicou a aquisição de conhecimentos de linguagens como o C# e o XML (*Extensible Markup Language*). Adicionalmente, foi necessária a familiarização com um dos softwares de programação de PLCs da Siemens utilizado na Consoveyo, S.A. – o TIA Portal®.

Paralelamente, iniciou-se o estudo dos sistemas desenvolvidos pela Consoveyo, S.A., tendo-se optado por desenvolver um novo programa de controlo para o *shuttle car*, dado que a implementação do TIA Portal Openness® obrigava à existência de um programa em TIA Portal®. O novo programa foi desenvolvido considerando a modularidade necessária à geração de código e tendo em conta, não só os princípios de funcionamento dos sistemas desenvolvidos pela empresa, mas também as variações mais comuns nas configurações dos mesmos.

Uma vez desenvolvido o programa e adquiridas as competências necessárias para a utilização do TIA Portal Openness® iniciou-se o desenvolvimento da aplicação, tendo por base a possibilidade de alterar o programa a partir da manipulação de documentos XML. A aplicação resultante manipula e gera o código do programa de controlo do *shuttle car* de acordo com a configuração desejada, dentro de uma gama limitada de variações.

Em seguida, na Figura 1, é apresentado um diagrama de Gantt com as tarefas realizadas e a respetiva duração em semanas.

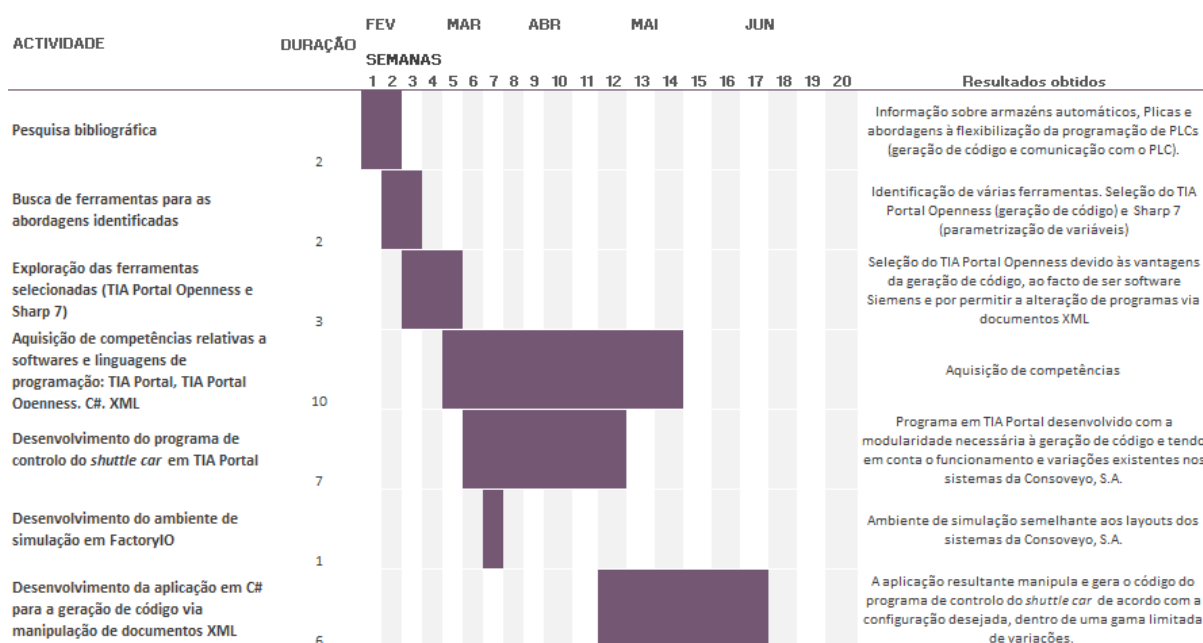


Figura 1 – Diagrama de Gantt: Tarefas realizadas ao longo da dissertação.

1.5 Estrutura da dissertação

O presente relatório encontra-se dividido em seis capítulos distintos, incluindo esta secção introdutória.

O capítulo 2 pretende servir de enquadramento teórico ao projeto desenvolvido, apresentando alguns aspetos importantes para a compreensão do mesmo. Neste, é feita uma revisão do estado da arte, abordando-se temáticas como os armazéns automáticos, os PLCs e, por último, a flexibilização da programação de PLCs, enfatizando-se a abordagem à geração automática de código.

No capítulo 3, são descritos os sistemas sobre os quais incide a presente dissertação: os armazéns automáticos e os *shuttle cars* concebidos pela Consoveyo, S.A.. Além disso, expõe-se o problema em causa, as abordagens investigadas e as ferramentas exploradas para cada uma delas. Por último, apresenta-se a abordagem selecionada para o desenvolvimento da aplicação de geração de código.

O capítulo 4, descreve o programa de controlo e o ambiente de simulação desenvolvidos, de modo a seguir a abordagem selecionada no capítulo anterior.

No capítulo 5, é apresentada a aplicação desenvolvida, a qual permite gerar os programas de controlo dos *shuttle cars* consoante as variantes a implementar. São ainda descritos os objetivos da aplicação, bem como a sua arquitetura e princípio de funcionamento.

No capítulo 6, é feita uma conclusão dos objetivos e resultados atingidos no decorrer desta dissertação, bem como uma perspetiva de desenvolvimentos futuros.

2 Estado da Arte

A geração de código para PLCs representa uma realidade promissora e inovadora. No entanto, e uma vez que se trata de um conceito relativamente recente, a informação disponível é bastante limitada.

Neste contexto, em primeiro lugar, optou-se por fazer uma breve exposição sobre armazéns automáticos, uma vez que o *shuttle car* alvo deste projeto é parte integrante de um destes sistemas. Em segundo lugar, faz-se uma breve introdução aos PLCs e à principal norma de referência para estes equipamentos (IEC 61131). Por último, abordam-se as alternativas existentes no que diz respeito à flexibilização da programação de PLCs.

2.1 Armazéns automáticos

Os sistemas de armazenamento automáticos têm vindo a ser amplamente implementados nas mais diversas áreas da indústria. A integração destes sistemas ao nível industrial permite reduzir a mão de obra necessária aos processos de armazenamento e recolha de produtos dos armazéns, possibilitando um consequente aumento da produtividade. Adicionalmente, estes sistemas conferem elevadas taxas de precisão e fiabilidade comparativamente aos sistemas manuais, sujeitos a erros humanos (Parsons *et al.* 1995).

Os armazéns desenvolvidos pela Consoveyo, S.A. podem integrar diversos equipamentos (Capítulo 1), entre os quais os *shuttle cars* ou carros de transferência (Figura 2), sobre os quais se foca a presente dissertação. Os *shuttle cars* da Consoveyo, S.A. são utilizados para transferir paletes entre diferentes linhas de transportadores do sistema. São veículos guiados por carris que dispõem de um ou mais transportadores e que recebem comandos por parte do PLC sobre a gestão da entrega e recolha de paletes, controlo da translação e do transportador, funções de segurança e diagnóstico, etc. (Consoveyo 2017c)

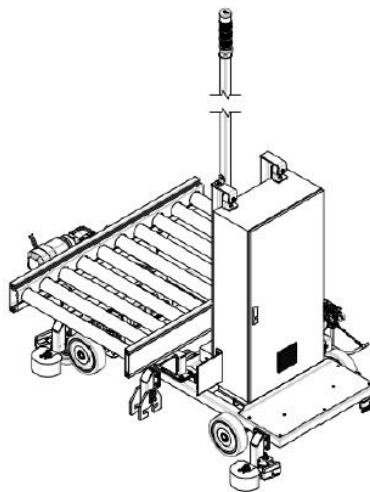


Figura 2 – Exemplo de *shuttle car* produzido pela Consoveyo. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

2.2 Controladores Lógicos Programáveis

Os PLCs são computadores industriais digitais concebidos de forma robusta e adaptada ao controlo de sistemas industriais, como linhas de montagem ou dispositivos robóticos. Estes diferenciam-se dos restantes sistemas computacionais por tolerarem condições de trabalho severas (humidade, temperatura, etc.), possuírem capacidades de tempo real e por disponibilizarem interfaces industriais (I/O digitais, redes de campo) (Alphonsus e Abdullah, 2016).

2.2.1 História dos PLCs

Estes equipamentos foram desenvolvidos pela indústria automóvel, com vista a substituir os circuitos cablados (com relés, temporizadores, etc.) por um sistema mais flexível. A sua implementação tornou possível a utilização do mesmo hardware em múltiplas aplicações, dada a capacidade de alterar rapidamente o comportamento de um circuito através do seu programa (IDC Technologies 2002).

Os primeiros PLCs eram dispositivos grandes, dispendiosos e com uma gama limitada de funcionalidades. Estes operavam apenas com *inputs* e *outputs* discretos, o que apenas permitia a sua aplicação em operações que requeressem movimentos repetitivos. Para além disso, a sua linguagem de programação era bastante complexa, requerendo programadores altamente qualificados (Hackworth e Hackworth, 2004).

A crescente implementação dos PLCs no controlo de sistemas industriais fomentou a sua evolução, tornando-os equipamentos cada vez mais versáteis. As inovações e melhorias nos microprocessadores e nas técnicas de programação tornou-os cada vez mais rápidos, e possibilitou a sua aplicação no controlo de processos e movimentos mais complexos (Alphonsus e Abdullah, 2016). Atualmente, são considerados equipamentos indispensáveis no controlo de sistemas industriais (IDC Technologies 2002).

2.2.2 Norma IEC 61131

A norma IEC 61131, é a principal norma de referência acerca dos PLCs e foi desenvolvida pela Comissão Eletrotécnica Internacional (IEC) no sentido de normalizar as diferentes visões e filosofias dos fabricantes de PLCs. A norma está dividida em diversas partes, entre as quais (International Electrotechnical Commission 2018):

1. Informação geral;
2. Requisitos de equipamento e testes;
3. Linguagens de programação;
4. Diretrizes para os utilizadores;
5. Comunicações;
6. Segurança funcional;
7. Programação *Fuzzy Control*;
8. Diretrizes para a aplicação e implementação de linguagens de programação;
9. Interface de comunicação digital *Single-Drop* para pequenos sensores e atuadores;
10. Formatos de intercâmbio XML para programas de acordo com a IEC 61131-3.

2.2.2.1 IEC 61131-3: Linguagens de programação

A programação dos primeiros PLCs era implementada em diagrama de contactos elétricos, mantendo-se até à atualidade como uma das linguagens de programação mais populares. No entanto, dada a enorme evolução dos PLCs, as capacidades básicas desta linguagem não eram suficientes, obrigando os fabricantes de PLCs a desenvolver novas ferramentas. Tal facto, resultou numa multiplicidade de linguagens de programação, o que obrigava ao uso do software proprietário de cada fabricante e complicava a aprendizagem das mesmas (IDC Technologies 2002).

No entanto, os fabricantes de PLCs rapidamente perceberam que o futuro destes equipamentos e seu amplo uso na indústria não seria possível, a menos que as questões da portabilidade e interoperabilidade fossem resolvidas (IDC Technologies 2002). Assim, iniciou-se um processo de uniformização da programação do qual resultou a norma IEC 61131-3. Esta prevê as seguintes linguagens de programação (International Electrotechnical Commission 2018):

- Textual
 - Lista de Instruções (IL)
 - Texto Estruturado (ST)
- Gráfica
 - Diagrama de Blocos (FBD)
 - Diagrama de Contactos Elétricos (LD)
- Sequential function chart (SFC)

2.2.2.2 IEC 61131-3: Arquitetura do software

A arquitetura de software descrita na norma IEC 61131-3 (Figura 3), foi desenvolvida com objetivo de uniformizar a abordagem na programação dos PLCs, prevendo o impacto que as tecnologias emergentes e os futuros requisitos de controlo terão nas estruturas de software e linguagens de programação dos PLC.

De acordo com a norma, de modo a que um programa seja executado por um PLC, este requer os seguintes tipos de interfaces:

- interfaces I/O: conexão do PLC ao meio físico;
- interfaces de sistema: serviços requeridos para assegurar a execução do programa;
- interfaces de comunicação: para comunicar com outros equipamentos, PLCs, interfaces homem-máquina (HMIs) e outros *outputs*;

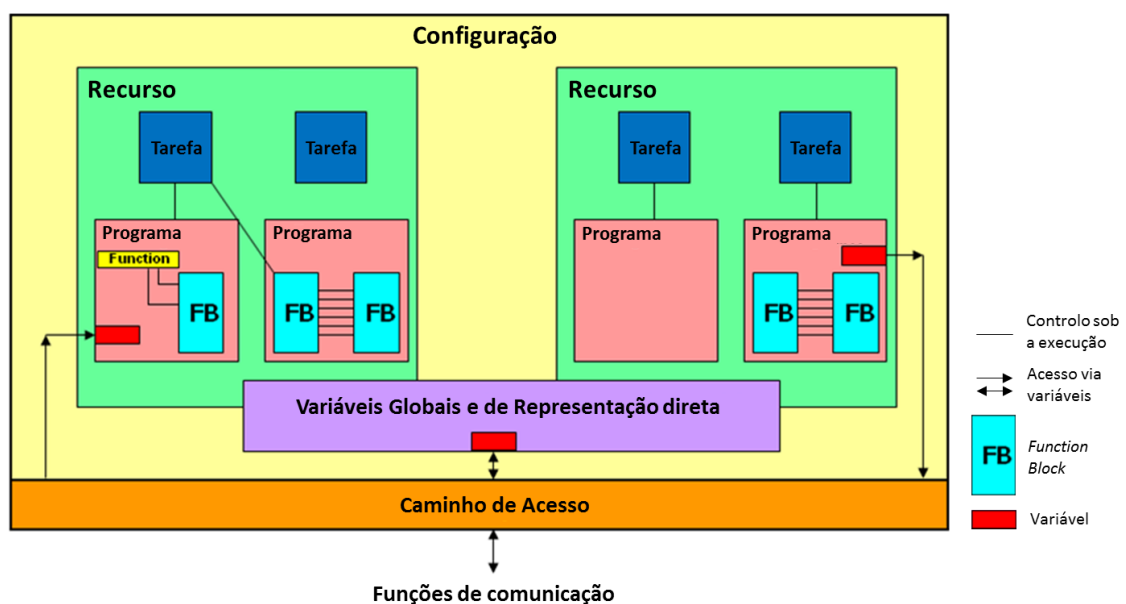


Figura 3 – Arquitetura do software descrita pela norma IEC 61131-3. In <http://ua.automation.com/resources-tools/application-stories/programmable-logic-controller-plc/iec-61131-3-functions-and-function-blocks-what-is-the-difference>, 11-06-2018, 16:52.

Como se pode constatar a partir da observação da Figura 3, o modelo de software contém um conjunto de componentes organizados hierarquicamente. Estes elementos, são sucintamente descritos em seguida.

- Configuração: Software contido num PLC de um dado sistema.
- Recurso: Suportam e ordenam a execução de um ou mais programas. Uma configuração pode conter múltiplos recursos.
- Tarefa: Elementos de software utilizados pelos recursos e que invocam um ou mais programas ou *function blocks*.
- Unidades de Organização do Programa: incluem programas, *function blocks* ou *functions*. Estes podem ser reutilizados na forma de instâncias.
- Programa: Construção lógica necessária ao processamento de sinais requeridos para o controlo de uma máquina ou processo.
- Function Block: Permite subdividir um problema complexo em problemas mais simples, de modo a facilitar a sua abordagem. Para além disso, podem ser aplicados em situações que necessitem da mesma lógica, a partir de uma instanciação com diferentes parâmetros. Os *function blocks* definem um conjunto *inputs*, *outputs* e variáveis internas, os quais são armazenados numa estrutura de dados, única para cada instância do mesmo.
- Function: Uma *function* tem diversos *inputs* a partir dos quais, após determinadas operações, fornece um único *output*. Ao contrário dos *function blocks* não possuem nenhum mecanismo de armazenamento.
- Variáveis locais e globais: As variáveis locais podem ser chamadas dentro de uma configuração, programa, *function block* ou *function*, dependendo do elemento de software no qual foram declaradas. Contrariamente, as variáveis globais podem ser acedidas a partir de qualquer elemento de software dentro desse programa.
- Variáveis de representação direta: são usadas para endereçar diretamente os locais da memória de um PLC.
- Caminho de Acesso: associação de um nome simbólico a uma variável para fins de comunicação.

2.3 Práticas aplicadas na Consoveyo, S.A.

2.3.1 Linguagens de programação

Na Consoveyo, S.A., a maioria da programação é implementada usando diagramas de contactos eléctricos, que, dada a sua semelhança a esquemas eléctricos, simplifica a compreensão da mesma. Simultaneamente, o seu uso permite uma mais fácil compreensão do código por parte de pessoal não programador, como técnicos fabris e outro pessoal que necessite de contactar com o código. Há também alguns troços de código que são programados com listas de instruções (IL).

2.3.2 Plataformas utilizadas: TIA Portal® e SIMATIC STEP 7®

O TIA Portal® e o SIMATIC STEP 7® são as plataformas utilizadas para a programação dos PLCs da Siemens utilizados nos sistemas desenvolvidos pela Consoveyo, S.A..

O SIMATIC STEP 7® é um pacote de software que permite programar os PLCs SIMATIC da Siemens, das séries 300 e 400. O pacote *standard* inclui *symbol editor*, *SIMATIC Manager*, *NETPRO Communication Configuration*, que permite fazer a configuração das redes, *Hardware Configuration*, que permite fazer a configuração do hardware a utilizar e

Hardware Diagnostics. Suporta linguagens de programação abordadas na norma IEC 61131-3, entre as quais: diagramas de contactos eléctricos, diagramas em blocos e listas de instruções. Atualmente, este é o software mais utilizado pela Consoveyo, S.A. no desenvolvimento dos seus projetos de automação, existindo, no entanto, como objetivo a médio prazo, fazer a migração dos mesmos para o TIA Portal® (Siemens 2017b).

O TIA Portal® é um pacote de software desenvolvido pela Siemens, que surgiu como resposta à crescente complexidade da realidade industrial. Representa uma inovadora e poderosa ferramenta de engenharia, permitindo integrar as mais diversas tarefas de automação. O TIA Portal® combina as funcionalidades do STEP 7®, possibilitando a programação dos PLCs SIMATIC da Siemens das várias séries nas cinco linguagens de programação descritas na norma IEC 61131-3, com as do WinCC®, que permite o *design* de HMIs, com a configuração de drives e controlo de movimento. Este programa possibilita também a configuração das redes de comunicação e do hardware de uma forma mais intuitiva do que o STEP 7® (Siemens 2017d).

2.4 Flexibilização da programação de PLCs

A generalidade das indústrias tem reformulado as suas estratégias de negócio, de modo a sobreviver ao ambiente extremamente competitivo vivido na atualidade. Esta mudança passa por torná-las estruturas mais ágeis, orientadas ao produto e aos requisitos do cliente. Para além disso, é necessário aumentar os valores da produtividade, não menosprezando a importância da qualidade dos produtos. As soluções automáticas são frequentemente prescritas como solução para alguns destes problemas. A sua implementação permite aumentar a cadência de produção e melhorar a qualidade dos produtos, possibilitando simultaneamente reduzir os custos com a mão de obra (Han 2010).

Fruto do referido ambiente competitivo e da crescente exigência das indústrias, os fornecedores de soluções automáticas debatem-se diariamente com novos desafios e requisitos impostos por parte dos seus clientes. A procura de soluções automáticas customizadas contrasta com a elevada pressão colocada sobre os custos e tempos de execução destes projetos. No entanto, os fornecedores deste tipo de soluções foram obrigados a adaptar-se às exigências do mercado e a desenvolver métodos para reduzir estas estatísticas.

Na conceção de um sistema automático, a fase de automatização domina grande parte da totalidade dos tempos e custos da implementação, pelo que é prática comum reutilizar soluções *standard* na programação destes sistemas. Atualmente, é maioritariamente aplicada pelos engenheiros segundo um processo manual. Dependendo da fase de implementação do projeto, soluções *standard* são arquivadas e geridas em bibliotecas, para uso posterior. A seleção das soluções apropriadas está dependente do conhecimento da aplicação em causa e da experiência adquirida por parte dos engenheiros. O inconveniente desta prática reside na carga de trabalho manual que se mantém significativamente elevada (Güttel, Weber e Fay, 2008).

Neste sentido, e dado que a maioria dos sistemas automáticos são controlados com recurso a PLCs, a comunidade científica e o mundo industrial têm procurado alternativas para flexibilizar a sua programação, e consequentemente agilizar a conceção de sistemas automáticos. De acordo com a bibliografia consultada, identificaram-se duas possíveis abordagens para lidar com esta problemática: a geração de código e a parametrização de variáveis a partir da comunicação com o PLC. Em seguida explora-se cada uma delas, enfatizando-se a geração de código, dado ser o tema da presente dissertação.

2.4.1 Geração automática de código

Uma possível abordagem à flexibilização da programação de PLCs é a geração automática de código (Güttel, Weber e Fay, 2008). A comunidade científica tem apresentado grandes vantagens relativamente à implementação desta abordagem, não havendo, até ao momento, nenhuma metodologia amplamente utilizada para a execução desta tarefa (Han 2010).

Segundo Steinegger *et al.* (2016), a existência de uma ferramenta que gere código para PLCs permite reduzir o esforço de programação e os erros de *design* introduzidos manualmente na implementação de aplicações de controlo. Além disso, possibilita reduzir os atrasos na fase da automatização do sistema ou programação, que podem levar a uma enorme perda de receita por parte dos fornecedores de sistemas automáticos (Thapa *et al.* 2009). Assim, o uso de uma ferramenta que permita uma rápida modelação e compilação, de modo a gerar código *standard* para PLC, é um método eficiente de reduzir o tempo de execução dos projetos e aumentar a produtividade e eficiência dos fornecedores de sistemas automáticos (Steinegger *et al.* 2016).

Ao contrário do que acontece com os PLCs, a geração de código para equipamentos como robôs industriais e máquinas de comando computadorizado (CNCs) é uma abordagem muito utilizada no mundo industrial, sendo implementada através de ferramentas chamadas pós-processadores.

No caso dos CNCs, os pós-processadores são softwares responsáveis por converter as trajetórias da ferramenta, geradas por um sistema de fabrico assistido por computador (CAM), em programas que possam ser lidos e executados pelo controlador da máquina. Uma vez concluída a modelação em CAD da peça, a trajetória da ferramenta é gerada por um sistema CAM, dando origem a um ficheiro *cutter-location (CL-file)*. Por último, este é pós-processado, através do modelo cinemático, obtendo-se um programa, conhecido por *G-code* (Baron 1999). Os programas *G-code* têm sido vastamente utilizados ao longo das últimas décadas e contêm a informação relativa ao posicionamento, velocidade dos motores e trajetória a cumprir (Xu e Newman, 2006).

2.4.1.1 Requisitos para a geração de código

Na geração automática de código é necessário considerar o comportamento desejado para o sistema para além da sua sequência nominal de funcionamento, isto é, para além do regime ideal de funcionamento do sistema (sem interrupções de segurança, fluxos de material, etc.). Assim, o desenvolvimento de uma ferramenta de geração de código exige a inclusão de funcionalidades exteriores a este regime, entre as quais funções de segurança, sequências de arranque e paragem, etc. Caso contrário, a ferramenta abrangerá apenas uma pequena parte das funcionalidades a serem implementadas no código, pelo que será necessária a posterior adição manual das restantes funcionalidades (Güttel, Weber e Fay, 2008).

Adicionalmente, e segundo Güttel, Weber e Fay (2008), o código gerado automaticamente deve cumprir determinadas condições, entre as quais:

- modularidade: composto por *function blocks* testados, os quais são incluídos numa biblioteca;
- transparência: ter uma estrutura definida, facilmente perceptível para o utilizador;
- expansividade: estruturado de forma a poder ser adaptado e expandido.

A possibilidade de gerar código para PLCs com a qualidade desejada depende, em grande parte, da disponibilidade de fontes de informação num formato legível por computador. Para além disso, de modo a atingir uma solução automática de geração de código, é necessário que o processo esteja bem definido e que apresente um elevado grau de uniformização. Deste

modo, segundo Güttel, Weber e Fay (2008), a criação de código para PLC deve cumprir quatro requisitos. Em primeiro lugar, a estrutura da instalação deve estar num formato legível pelo computador, de modo a possibilitar, por exemplo, a interligação entre um *conveyor* e os seus sensores correspondentes. O segundo requisito refere-se à descrição do processo num formato legível pelo computador, que, segundo Güttel, Weber e Fay (2008), deve ser feita usando a linguagem SFC da norma IEC 61131-3 e representada em formato PLCOpen-XML. Em terceiro lugar é fundamental dispor de uma biblioteca de *function blocks* bem definida e uniformizada. Estes blocos são de extrema importância pois compreendem as funcionalidades respetivas aos vários componentes do sistema. Por último, o conhecimento proveniente da experiência por parte do engenheiro do processo deve ser transferido para um formato legível pelo computador.

2.4.1.2 Casos de estudo

Como foi referido ao longo desta exposição, a geração automática de código para PLCs é uma temática recente e inovadora, o que dificultou a identificação de publicações científicas que revelassem metodologias para lidar com esta problemática.

Diversas fontes (Frey e Litz (2000), Thapa *et al.* (2009) e Hasdemir e Kurtulan (2006)) citam a utilização de métodos formais como modo de abordar a geração automática de código. Outros autores (Thramboulidis e Frey (2011) e Thramboulidis (2012)) referem a utilização de linguagens de modelação, segundo um processo de engenharia orientada a modelos, como uma possível abordagem à geração de código. No entanto, estas fontes não exploram, ao pormenor, a forma como é feita a efetiva geração de código.

Outra possibilidade foi sugerida por Andersson e Hellander (2010), que revela a possibilidade de flexibilizar a programação de linhas de produção com recurso ao software *Automation Designer*® da Siemens. No entanto, não foi possível obter este software de modo a poder explorar as suas potencialidades.

Métodos Formais

Os métodos formais como a teoria dos autómatos, as redes de petri ou t-MPSG permitem fazer a especificação formal do controlador a partir da descrição informal do processo não controlado e dos requisitos do sistema controlado (Figura 4 – percurso interior) (Thapa *et al.* 2009). Apesar destes métodos não serem uma constante na realidade industrial, devido à sua complexidade, segundo Frey e Litz (2000), a programação de PLCs será cada vez mais suportada por métodos formais. Tal facto deve-se à crescente complexidade dos problemas de controlo e à necessidade de reduzir tempos de desenvolvimento e de reutilizar módulos de software. Para além disso, a exigência de soluções com cada vez mais qualidade e, especialmente, a aplicação de PLCs em sistemas críticos, torna necessária a utilização de procedimentos de verificação e validação, isto é, métodos formais. Na indústria, é prática habitual fazer a implementação direta do controlador usando uma linguagem de programação de PLCs para chegar à fase de realização (Figura 4 – percurso exterior), que inclui o hardware e o software (Frey e Litz, 2000). Assim, o *design* lógico é geralmente omitido na programação dos PLCs, apesar da sua importância na deteção de erros lógicos previamente à implementação (Han 2010). Posteriormente, recorre-se a um processo de validação informal, que consiste no confronto dos resultados dos testes do controlador implementado com a especificação informal do sistema. Este processo é bastante moroso, não conseguindo ser totalmente completado (Frey e Litz, 2000). No entanto, para o caso das indústrias de desenvolvimento de sistemas críticos, como é o caso das indústrias aeronáutica, aeroespacial e ferroviária existe regulamentação que obriga à validação e verificação do código-fonte a partir de métodos formais. Esta atividade é

geralmente desenvolvida por empresas independentes como a Critical Software (Critical Software 2018).

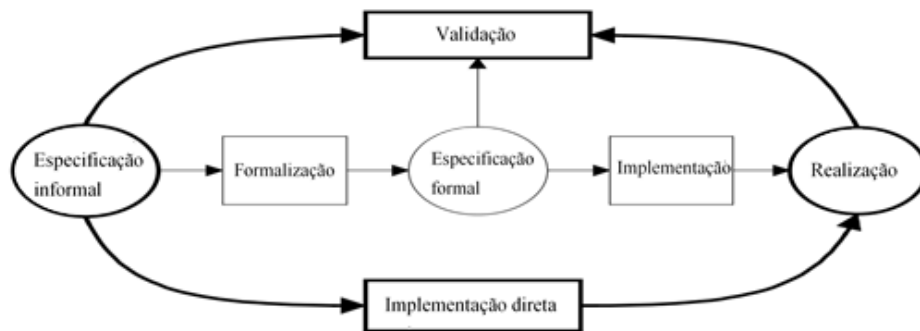


Figura 4 – *Design* do processo para sistemas de controlo lógicos (percurso exterior: *design* comum praticado pelas indústrias; percurso interior: *design* segundo métodos formais). In Frey e Litz, *Formal methods in PLC programming*, 2000.

Engenharia Orientada a Modelos

A Engenharia Orientada a Modelos (*Model Driven Engineering* – MDE) é uma metodologia de desenvolvimento de software com relevante sucesso na generalidade das aplicações de computação. Este facto tem levado à exploração dos benefícios da sua implementação no domínio da automação industrial (Thramboulidis e Frey, 2011).

Num processo de MDE são criados modelos conceptuais que permitem uma visão geral e abstrata de um dado sistema ou problema. O primeiro a ser desenvolvido é o modelo independente da plataforma (*Platform Independent Model* - PIM). O PIM é uma representação de alto nível de um software, independente de qualquer tecnologia de implementação. Através de uma linguagem de transformação de modelos (MTL) é possível transformar o PIM num modelo específico da plataforma (PSM). Este descreve a forma a partir da qual o PIM é implementado numa dada plataforma ou tecnologia (Thramboulidis 2012) (Thramboulidis e Frey, 2011).

Deste modo, segundo Thramboulidis e Frey (2011) e Thramboulidis (2012) é possível gerar código para PLCs a partir de um processo de MDE baseado na norma IEC 61131, utilizando diagramas de instrumentação e linguagens de modelação como o UML ou o SysML.

Geração automática de programas para PLC via *Automation Designer*

Segundo Andersson e Hellander (2010), a redução do tempo de execução de sistemas automáticos de produção passa frequentemente pela execução de simulações virtuais como forma de investigar o comportamento do sistema. Para a implementação da simulação é necessária a totalidade lógica, não podendo esta, no entanto, ser transferida diretamente para o PLC. Tal facto, faz com que seja necessária uma transferência manual, resultando em trabalho em duplicado. Utilizando o software *Automation Designer*® da Siemens, foi possível reutilizar informação de projetos anteriores de modo a poder aplicá-la em novos projetos. Foram criados *templates* reutilizáveis, contento informação que se mantinha invariável nos vários projetos. Através da adição de *function blocks*, disponibilizados numa biblioteca, aos diversos *templates*, foi gerado código para o software SIMATIC STEP 7® da Siemens, bem como o hardware e a *symbol table* necessários ao projeto. Este software, além de possibilitar a geração de programas para PLCs, permite ainda a geração de HMIs e de esquemas elétricos, a partir da importação de ficheiros XML para os softwares WinCC® e EPLAN Electric P8®, respetivamente. A interface

criada é constituída por um projeto base, que atua como uma base de dados, na qual são criados objetos (*conveyors*, robôs, etc.), caracterizados por atributos, e *templates*. Este projeto base pode ser encarado como um modelo a partir do qual novos projetos são construídos. Em cada projeto interligam-se objetos e *templates*, de modo a obter sistemas mais complexos. Ao inserir o mesmo objeto em diferentes *templates* torna-se possível configurar os seus atributos de formas distintas, possibilitando múltiplas funcionalidades. Os autores ressaltam que, de modo a poder implementar este software, é de extrema importância utilizar ou, se necessário, criar, uma estrutura padrão para os projetos.

2.4.1.3 Ferramentas existentes

Através da consulta bibliográfica efetuada identificaram-se diversas ferramentas que permitiam abordar a problemática da geração de código para PLCs. No entanto, para a maioria delas, não existe informação em abundância acerca do seu funcionamento e potencialidades. Para além disso, as ferramentas identificadas são bastante recentes, não se tendo encontrado nenhuma aplicação que refira a sua utilização.

TIA Portal Openness®

O TIA Portal Openness® é uma extensão incluída nos pacotes de instalação do software TIA Portal®. Esta ferramenta tem como objetivo reduzir o trabalho redundante atuando como gerador de código, através da sua interface no Microsoft Visual Studio®. Possibilita uma maior competitividade, precisão e eficiência, uma vez que permite uma execução rápida das tarefas de forma consistente (Siemens 2016). A utilização desta ferramenta implica conhecimentos básicos sobre programação orientada a objetos, ambiente de programação do Microsoft Visual Studio® e configurações do TIA Portal®. De acordo com a informação disponibilizada pela Siemens, esta ferramenta possibilita um vasto leque de funções, de entre as quais: comunicação com o TIA Portal®, exportação/importação de módulos de PLCs, tabelas de variáveis (*tagtables*) e blocos de funções (*function blocks*), para além de diversas funções vocacionadas para HMIs (Siemens 2017a).

Simulink PLC Coder®

O Simulink PLC Coder® permite fazer geração de código para PLCs em diagramas de contactos elétricos (LD) ou texto estruturado (STL) em acordo com a norma IEC 61131-3 independentemente do hardware, a partir de modelos *Simulink*, gráficos *Stateflow* e funções MATLAB®. O código é gerado em PLCOpen-XML ou noutros formatos suportados por ambientes de desenvolvimento integrado (IDEs) amplamente utilizados, incluindo 3S-Smart Software Solutions CODESYS®, Rockwell Automation Studio 5000®, Siemens TIA Portal® e Omron Sysmac Studio®. Deste modo, é possível compilar e implementar a aplicação em vários PLCs (Mathworks 2017). Esta aplicação parece ser uma das mais maduras ferramentas de geração de código. No entanto, a utilização de gráficos *Stateflow* é propensa a erros, uma vez que têm uma semântica que não se aplica ao domínio dos PLCs. Mesmo para programadores experientes, esta ferramenta pode gerar dúvidas e problemas (Darvas, Vinuela e Majzik, 2016).

CIF 3 PLC Code Generator®

O gerador de código CIF 3 PLC code generator® possibilita a geração de código para PLCs segundo a norma IEC 61131 (Universidade Técnica de Eindhoven 2014). Segundo

Darvas, Vinuela e Majzik (2016), esta ferramenta académica é promissora, sendo, no entanto baseada numa linguagem de modelação textual demasiado complexa e matemática, tornando necessária muita prática com a aplicação.

2.4.2 Parametrização de código a partir da comunicação com o PLC

A comunicação com o PLC pode ser conseguida através de diversas ferramentas que implementam protocolos industriais de comunicação. Este tipo de ferramentas facilita a troca de informação entre PLCs e computadores, permitindo implementar diversas funções, entre as quais a leitura e escrita de variáveis do e para o PLC, respetivamente. Apesar destas ferramentas serem preferencialmente utilizadas em aplicações de supervisão, nas quais é necessário adquirir sinais da instrumentação de campo e responder adequadamente, e na construção de HMIs, podem também ser utilizadas para flexibilizar a programação de PLCs. Esta abordagem, embora pouco documentada, é mais comum no meio industrial, permitindo a modificação de um programa a partir da alteração de valores de variáveis, o que pode possibilitar, por exemplo, seleccionar o código a ser executado pelo PLC.

2.4.2.1 Ferramentas existentes

Em seguida apresentam-se algumas ferramentas que permitem fazer a comunicação com PLCs da Siemens.

Snap7 Library

O Snap7 é uma biblioteca de funções *open source* que permite a comunicação com PLCs da Siemens das séries 300, 400, 1200 e 1500. Suporta diversas linguagens de programação, como por exemplo C++ e C#, e é multiplataforma, podendo trabalhar com diversos sistemas operativos e hardwares (Mesta Automation 2016). O Sharp7, a variação do núcleo do Snap 7 para C#, constitui um ficheiro .cs, ou seja, uma classe passível de ser importada para o *Microsoft Visual Studio* e que contém diversas funções, entre as quais: comunicação com o PLC, leitura e escrita de entradas, saídas, temporizadores, contadores e *database blocks* (DBs) e leitura do estado do PLC (Snap7 2016).

S7 Library

O S7 é uma ferramenta semelhante ao Snap7, permitindo a comunicação com o PLC e leitura/escrita de entradas, saídas e DBs, através de uma classe de funções desenvolvida em C#. No entanto, apesar de ser uma ferramenta *open source*, a documentação facultada é muito escassa (Mesta Automation 2015).

3 Sistema, problema em causa e ferramentas exploradas

Neste capítulo é feita uma breve abordagem aos componentes e princípios de funcionamento de um armazém automático produzido pela Consoveyo, S.A.. São também expostas algumas considerações acerca do funcionamento do *shuttle car* e do seu programa de controlo.

Seguidamente o problema em estudo é definido de uma forma mais concreta e enquadrada no contexto real, e são exploradas, em maior profundidade, ferramentas para cada uma das abordagens identificadas na Secção 2.4.

Por último é definida a abordagem a seguir na execução da presente dissertação.

3.1 Descrição geral do sistema

Na Figura 5, encontra-se disposto o *layout* geral de um armazém desenvolvido pela Consoveyo, S.A.. Este armazém é composto por duas áreas: área de produção, responsável por receber e encaminhar paletes de produto acabado e em curso de fabrico desde a zona de entrada à zona de armazenamento, e pela saída de produtos em curso de fabrico; e área de expedição, responsável pelo transporte de paletes de produto acabado desde a zona de armazenamento até à zona de expedição (Consoveyo 2017c).

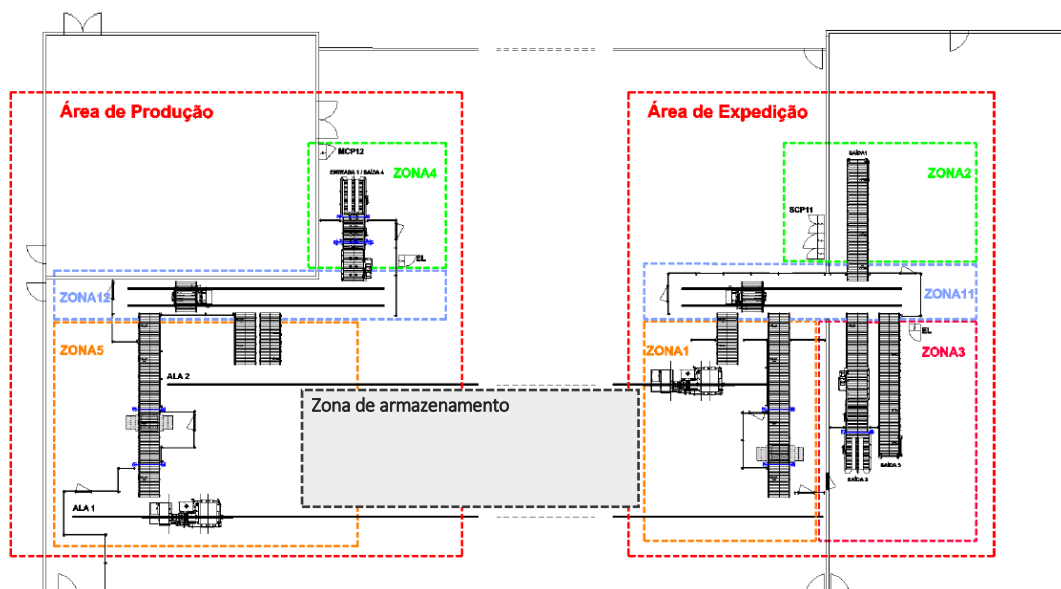


Figura 5 – *Layout* geral de um armazém automático. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

O sistema de periféricos do armazém automático disposto na Figura 5 encontra-se dividido em sete zonas funcionais. As áreas de produção e expedição englobam, respetivamente, as áreas 4,5 e 12 e 1,2,3 e 11. Apresenta-se em seguida uma breve descrição da totalidade das zonas (Consoveyo 2017c):

- **Zona 1** (Figura 6a): Engloba duas linhas de transportadores que servem de interface entre os transelevadores e o *shuttle car* na área de expedição;
- **Zona 2** (Figura 6b): Dotada de uma única linha de transportadores, permite encaminhar as paletes desde o *shuttle car* até aos cais de expedição número 1;

- **Zona 3** (Figura 6c): Possui duas linhas de transportadores que fazem a interface entre o *shuttle car* e os cais de expedição 2 e 3. Destaca-se o facto de o carregamento das paletes ser feito a cotas distintas nas duas linhas, devido à presença de um mini-elevador na linha 4;
- **Zona 4** (Figura 7a): Composta por uma linha de transportadores bidireccional responsável pelo controlo dimensional, pela receção e transporte de paletes desde a entrada até ao *shuttle car*, bem como pela saída de produtos em curso de fabrico provenientes da ala 2 e produtos rejeitados;
- **Zona 5** (Figura 7b): Dotada de três linhas de transportadores de interface com os transelevadores de ambas as alas. Os transportadores RC12.0201 e RC12.0202 e RC12.0301 a RC12.0304 constituem duas linhas de entrada para o armazém. Por sua vez, os transportadores RC12.0203 e RC12.0204 constituem uma linha de saída de produto em curso de fabrico;
- **Zona 11** (Figura 8): Delimita o espaço de operação do *shuttle car* da área de expedição. Este carrega as paletes provenientes das alas 1 e 2 através dos transportadores de entrega RC11.0102 e RC11.0204, respetivamente, e descarrega-as num dos transportadores de destino RC11.0301, RC11.0401 ou RC11.0501.
- **Zona 12** (Figura 9): Envolve o espaço de operação do *shuttle car* da área de produção. Este carrega as paletes provenientes da área de produção e da ala 2, através do transportador do mini-elevador (RC 12.8102), e do transportador de entrega RC12.0204, respetivamente. Estas podem ser descarregadas nos tapetes de destino RC12.0201 e RC12.0301 para serem encaminhadas para o armazém e no mini-elevador para se dirigirem novamente à área de produção (no caso de um produto em curso de fabrico).

O sistema pode funcionar em dois modos: automático, no qual o PLC gere todo o sistema, recolhendo e encaminhando as paletes para os seus destinos; manual, para manutenção ou ajuste de posicionamento de cargas.

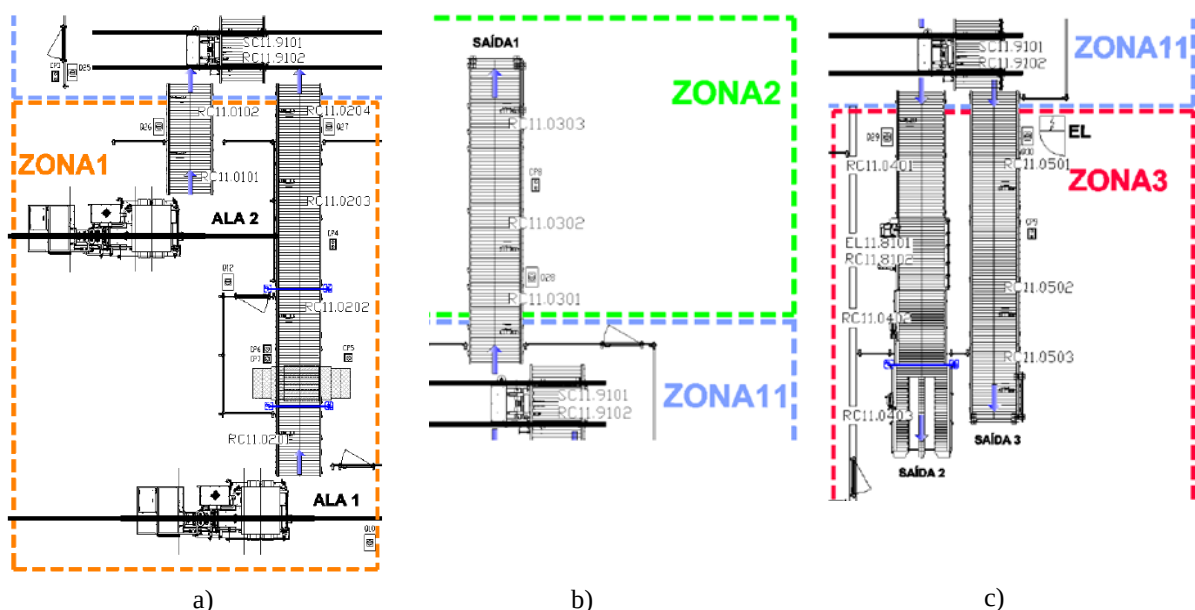


Figura 6 – Zonas funcionais 1, 2 e 3: a) Zona 1; b) Zona 2; c) Zona 3. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

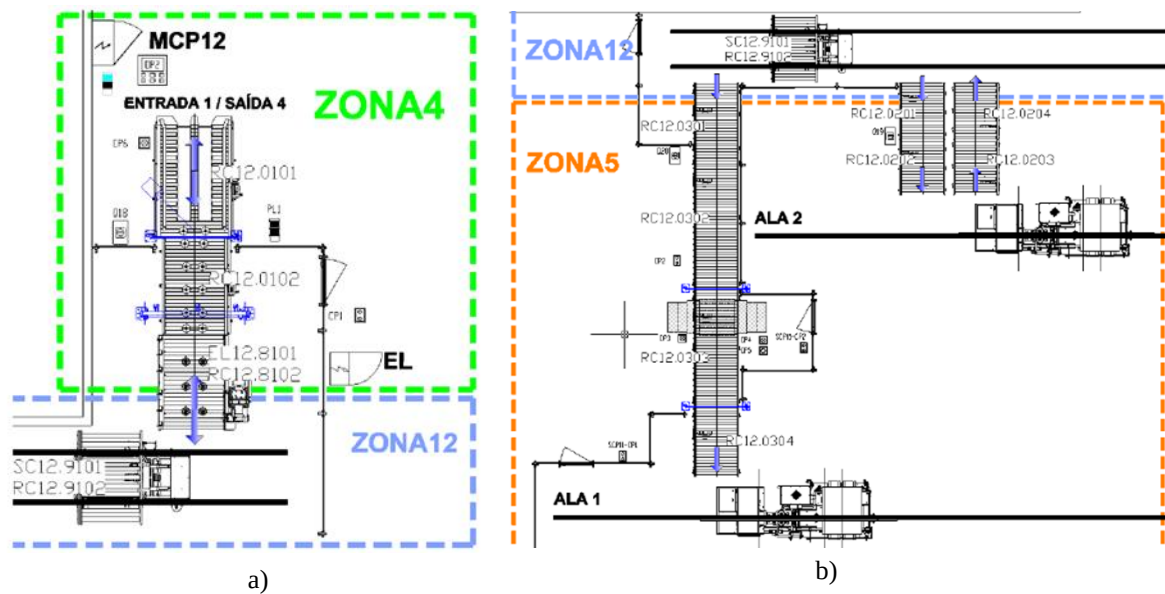


Figura 7 – Zonas funcionais 4 e 5: a) Zona 4; b) Zona 5. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

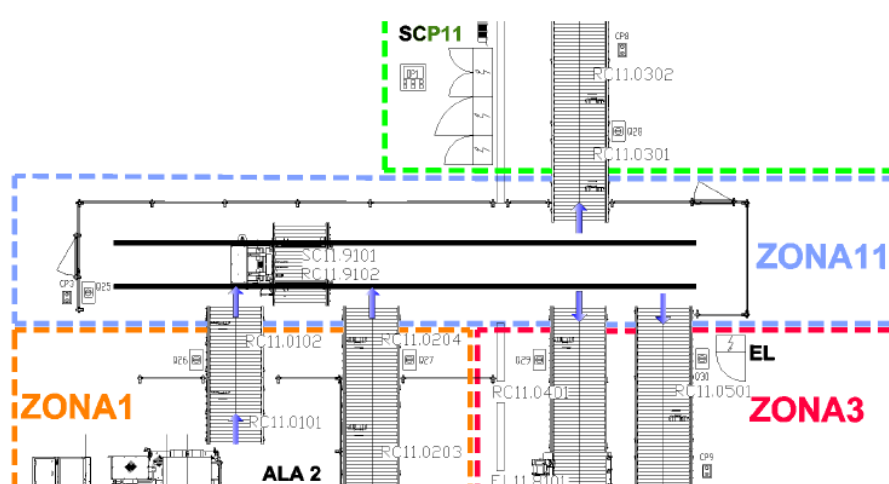


Figura 8 – Zona funcional 11. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

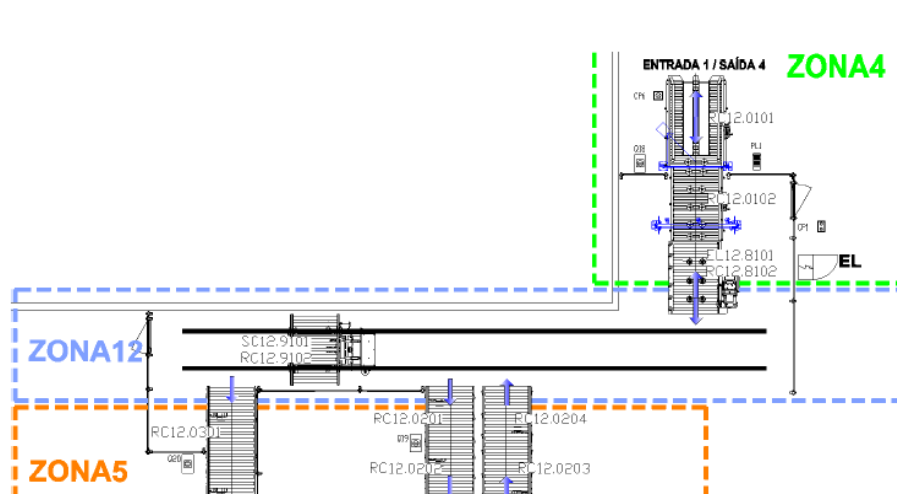


Figura 9 – Zona funcional 12. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

3.2 Princípio de funcionamento do *shuttle car*

Os *shuttle cars* ou carros de transferência, tal como descrito Capítulo 2, têm como finalidade transferir paletes entre diferentes linhas de transportadores. A carga é introduzida no sistema, em paletes, através das linhas de entrada (compostas por diversos transportadores) e transportada pelo *shuttle car* até à linha de destino desejada.

3.2.1 Constituintes e controlo do *shuttle car*

O *shuttle car* é composto por diversos constituintes, os quais se encontram identificados na Figura 10 (Consoveyo 2017c).

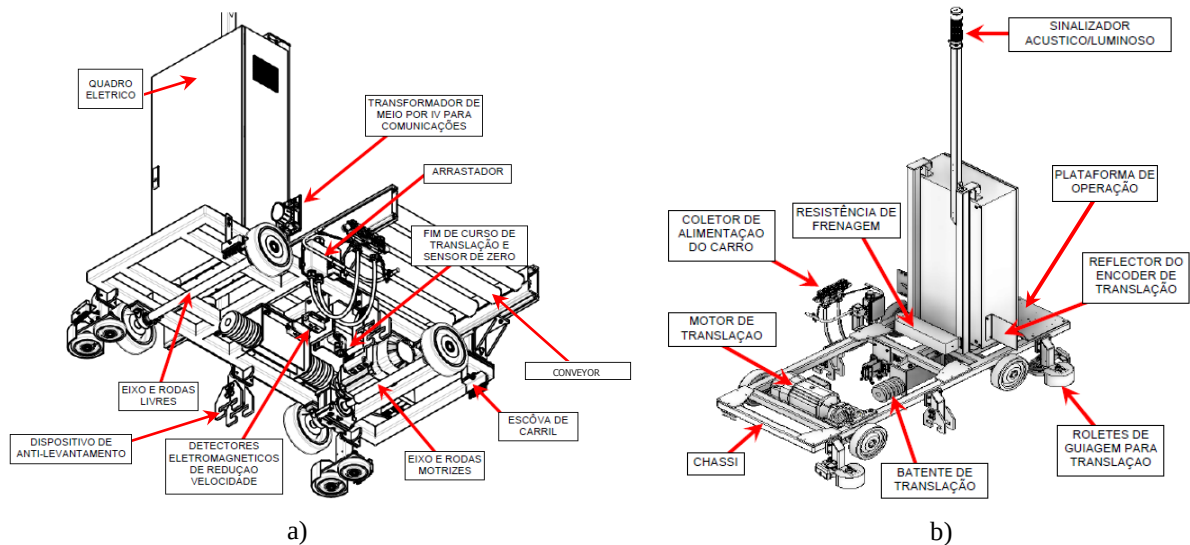


Figura 10 – Constituintes de um *shuttle car* produzido pela Consoveyo: a) vista de baixo; b) vista de frente. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

Como se pode visualizar, o *shuttle car* é composto por um sistema de comunicação por infravermelhos (Figura 10a), que transmite dados, entre o PLC do sistema de periféricos, que controla o *shuttle car*, e o PLC incorporado no seu quadro elétrico. O PLC do sistema de periféricos executa diversas funções, entre as quais (Consoveyo 2017c):

- Supervisão dos variadores de velocidade que controlam os movimentos de translação do *shuttle car* e do seu *conveyor*;
- Supervisão do circuito de emergência e deteção de falhas;
- Leitura dos sinais provenientes das fotocélulas, sensores e interruptores;
- Interface com o operador através do painel de controlo.

O dispositivo de medição laser, instalado no exterior da ala, permite determinar a distância ao refletor do encoder (Figura 10b) sem qualquer contacto físico e assim efetuar a avaliação e controlo do posicionamento do *shuttle car*.

Podemos ainda observar a presença dos detetores de redução de velocidade (Figura 10a). Estes são implementados próximo do final da ala do *shuttle car* com o objetivo de o abrandar antes deste atingir os limites de software (ordem de paragem) definidos no programa de controlo. Assim o *shuttle car* cessa o seu movimento antes de atingir os batentes ou fins de curso.

3.2.2 Modos de Operação

O *shuttle car* pode operar segundo dois modos distintos, seleccionados a partir da combinação dos seletores dos quadros elétricos da ala e do *shuttle* (Figura 11): automático, com as chaves do *shuttle* e da ala em automático e remoto, respetivamente; e manual, com as chaves do *shuttle* e da ala em manual e local, respetivamente (Consoveyo 2017c).

3.2.2.1 Modo Automático

No modo automático o *shuttle car* recebe ordens de um gestor. Este analisa a informação física (sensores) e lógica (ID paleta, destino e outros parâmetros) e determina em cada momento o posto a ser atendido e o destino pretendido para a paleta. Caso haja mais do que uma paleta a transportar é utilizado um tempo ponderado tendo em conta o comprimento do buffer. Em último caso a paleta a ser recolhida é aquela que se encontrar mais próxima do *shuttle car*.

A geração de uma ordem por parte do gestor é apenas possível caso se verifiquem todas as condições para a carga e descarga de uma paleta por parte do *shuttle car*, nomeadamente a presença de uma paleta no *conveyor* de entrega e a validade do destino final (disponibilidade do *conveyor* de destino). Uma vez dada a ordem, o *shuttle car* desloca-se para a coordenada de carga, disponibilizada pelo gestor, e verifica o seu posicionamento. Em seguida, inicia-se a transferência da paleta até que os sensores confirmem a seu correto posicionamento, isto é, que a paleta está centrada no *conveyor* do *shuttle car*. Após o carregamento, o *shuttle car* desloca-se para a coordenada de descarga verificando o seu posicionamento. Finalmente, inicia-se a transferência para o transportador de destino até que a paleta seja entregue corretamente.

3.2.2.2 Modo Manual

A movimentação do *shuttle car* em modo manual é conseguida através do painel de controlo incorporado no mesmo. É possível controlar o movimento de translação do *shuttle car* e do seu *conveyor* premindo continuamente as botoneiras “Translação” e “Mover Transportador”, respetivamente (Figura 11). A seleção da velocidade é conseguida através das botoneiras “Movimento +” e “Movimento –”. Premindo estas botoneiras a velocidade é incrementada e decrementada, respetivamente, até três vezes. Durante todas as operações no modo manual é necessário premir continuamente a botoneira de “Homem morto”, caso contrário a ação é interrompida.

No modo manual é ainda possível, através do seletor de bypass, ter dois modos de funcionamento distintos: com o seletor na posição “0”, a atuação dos fins de curso ou a abertura de outros circuitos de emergência torna impossível energizar os motores; com o seletor na posição “By-Pass”, todas as emergências deixam de ter interferência na operação do equipamento, podendo-se, por exemplo, retirá-lo dos limites da ala.

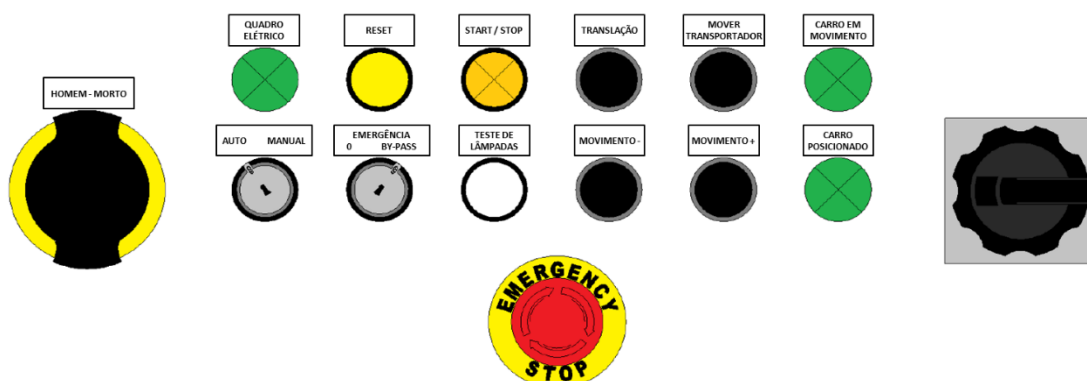


Figura 11 – Painel de controlo do *shuttle car*. In Consoveyo, Manual de Operação do Sistema de Periféricos, 2017

3.2.3 Variantes possíveis

Como referido no Capítulo 1, a Consoveyo, S.A. projeta armazéns automáticos de acordo com os requisitos do cliente, pelo que se torna difícil prever todas as variações possíveis no desenvolvimento dos mesmos. No entanto, no que diz respeito aos *shuttle cars*, existe um conjunto de variações mais comuns, indicadas em seguida:

- Número e posicionamento das linhas de carga e descarga;
- Número de *conveyors* por linha;
- Velocidades e acelerações;
- Número de *conveyors* do *shuttle car*;
- Tipo de *conveyor*: de rolos/correntes, com/sem elevação e com/sem inclinação;
- Integração de garfos em vez de *conveyor*;
- Integração de *conveyor* de interface (situação de *conveyor* de destino afastado);
- Número, tipo e posicionamento dos sensores.

3.3 Definição do problema e abordagens

Face ao elevado número de variações possíveis de implementar num *shuttle car* da Consoveyo S.A, torna-se impossível utilizar o mesmo programa de controlo para todos os projetos deste equipamento, sendo necessário adaptar manualmente o referido programa à situação em causa. Esta tarefa é bastante morosa, trabalhosa e suscetível a erros, pelo que seria ideal encontrar uma solução que permitisse a adaptação automática do programa de controlo às variações requeridas em cada novo projeto. É neste sentido que se enquadra o tema desta dissertação, que tem como objetivo final desenvolver uma aplicação de software que permita gerar o programa de controlo dos *shuttle cars* consoante as variações implementadas e assim flexibilizar e agilizar a programação destes equipamentos.

Apesar do tema da presente dissertação passar pelo desenvolvimento de uma aplicação de software para a geração automática de código para PLCs, o objetivo principal é a flexibilização da sua programação. Deste modo, exploraram-se ambas as abordagens referidas no Capítulo 2. A primeira, consiste na geração e manipulação de código para PLC, consoante as variantes e parâmetros escolhidos, a partir de uma aplicação externa. Embora, até ao momento, seja uma alternativa pouco explorada e, por isso, mais complexa, permite uma elevada flexibilidade na programação e pode trazer grandes vantagens no que diz respeito ao peso do programa de controlo resultante. Isto porque, utilizando esta técnica de geração *offline*, o programa final contém apenas o código necessário ao controlo do sistema. A segunda abordagem assenta na parametrização de variáveis através da comunicação com o PLC e consequente alteração do comportamento do programa. Esta alternativa, é limitadora para o futuro e gera um programa de controlo mais pesado e um código mais difícil de manter. Para além disso, a natureza *online* desta alternativa requer a presença do PLC em modo de execução.

De modo a explorar cada uma das abordagens, selecionaram-se duas das ferramentas descritas na Secção 2.4, uma para cada abordagem, com base na análise da sua viabilidade, potencialidades e compatibilidade com as ferramentas utilizadas na Consoveyo, S.A..

3.4 Ferramentas exploradas

Nesta secção são exploradas em profundidade as duas ferramentas seleccionadas, que vão de encontro a cada uma das abordagens referidas na secção anterior. Em primeiro lugar será explorado o TIA Portal Openness®, que permite abordar a problemática da geração de código. Posteriormente, avaliar-se-ão as potencialidades do Sharp7, que permite fazer parametrização de variáveis através da comunicação com o PLC

3.4.1 TIA Portal Openness

O TIA Portal Openness® é uma interface de programação de aplicações (*Public API* - Figura 12a) para o TIA Portal®. Permite criar aplicações próprias, a partir de um ambiente de desenvolvimento externo, como é o caso do Microsoft Visual Studio®. Segundo a Siemens, o TIA Portal Openness® permite automatizar diversas tarefas de engenharia, como por exemplo a programação, sendo possível desenvolver um gerador de código para HMIs e blocos de software para PLCs (Siemens 2018). Esta interface assenta na utilização de duas bibliotecas de vínculo dinâmico (DLL) (Siemens.Engineering.dll e Siemens.Engineering.HMI.dll - Figura 12b) que possibilitam a implementação de diversas funções de interface com o TIA Portal® (Siemens 2016).

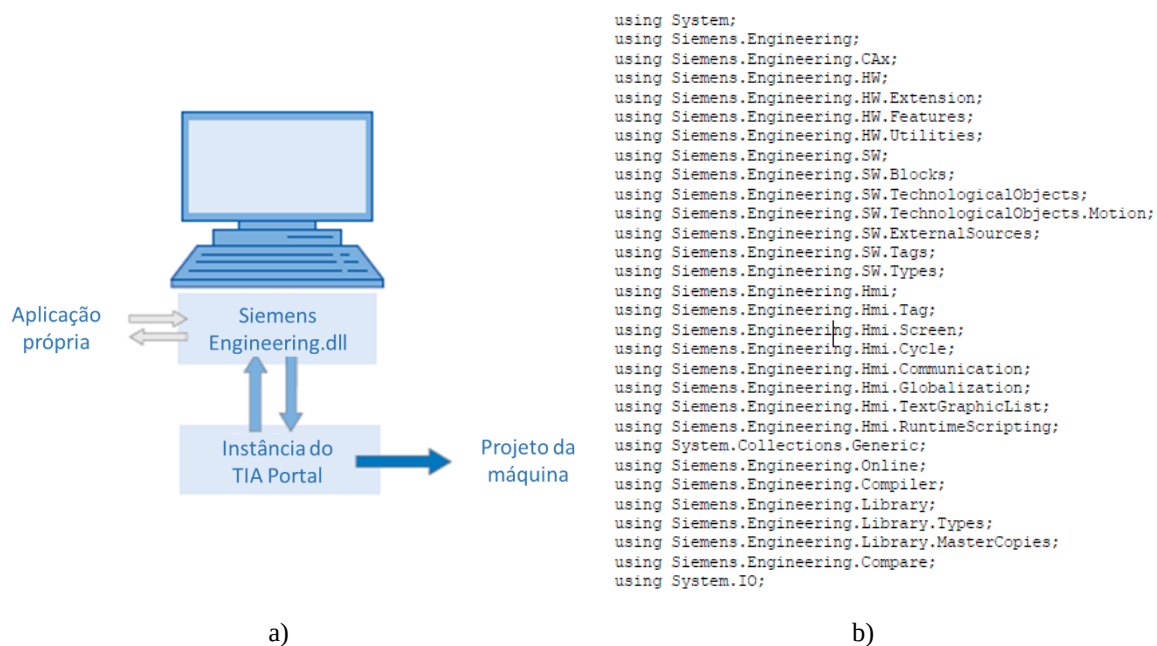


Figura 12 – Public API: a) Arquitetura; b) Bibliotecas incluídas. In Siemens, Automating projects with scripts, 2017

Para a instalação e gestão dos direitos de utilizador do TIA Portal Openness®, bem como para a inclusão dos ficheiros DLL no ambiente de desenvolvimento externo, recomenda-se a consulta dos procedimentos indicados no capítulo quarto do manual “TIA Portal Openness: Introduction and Demo Application” (Siemens 2016).

3.4.1.1 Funcionalidades

O TIA Portal Openness® permite aceder a diversas áreas do TIA Portal® a partir da interface *Public API*, nomeadamente: dados de projeto, PLCs e HMIs. Deste modo é possível implementar funções para cumprir determinadas tarefas, designadamente (Siemens 2017a):

- Aceder ao TIA Portal®, quer por criação externa de uma instância do TIA Portal® (com ou sem interface de utilizador), quer por acesso a instâncias a decorrer;
- Fechar, abrir e guardar um projeto;
- Enumerar, procurar, criar, modificar e apagar diversos objetos.

Para uma melhor compreensão da forma como é realizado o acesso às diversas áreas e objetos do TIA Portal®, a Siemens disponibiliza uma série de modelos que descrevem a organização da interface *Public API* (Siemens 2017a).

Para além das funcionalidades enumeradas anteriormente, o TIA Portal Openness® permite importar e exportar, a partir de documentos XML, a seguinte informação (Siemens 2017a):

- dados de projetos (*project graphics* e *project texts*);
- dados de HMIs (*tag tables*, *screens*, etc.);
- dados de PLCs (vários tipos de *program blocks*, *tag tables*, etc.);
- dados de hardware.

Deste modo, segundo a Siemens, o TIA Portal Openness® possibilita trocas de informação e fornecimento de dados de projetos para aplicações externas. Para além disso, o TIA Portal Openness® permite processar, externamente, dados de configuração para novos projetos com base em configurações existentes e importá-los posteriormente (Siemens 2017a).

3.4.1.2 Configurações possíveis

Segundo a Siemens, o TIA Portal Openness® pode operar segundo duas configurações distintas esquematizadas na Figura 13 (Siemens 2017a).

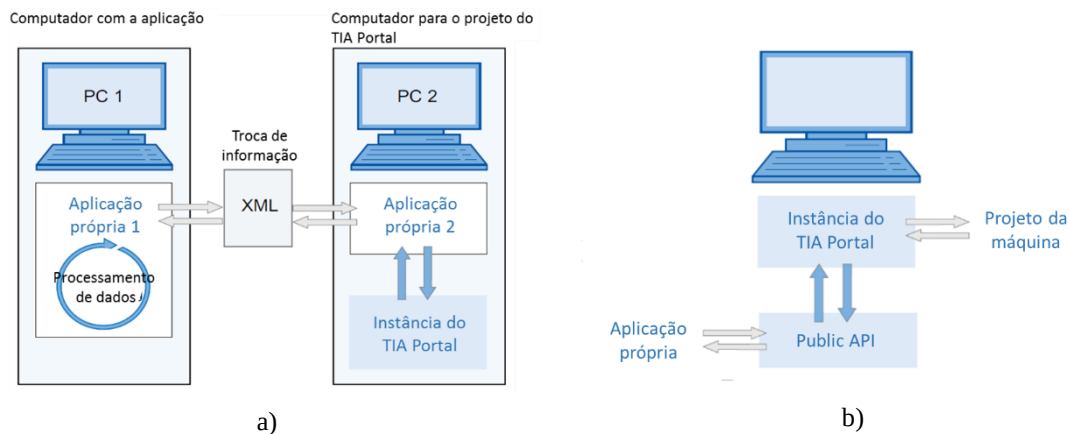


Figura 13 – Configurações do TIA Portal: a) Configuração de PCs distintos; b) Configuração de PC único. In Siemens, Automating projects with scripts, 2017

Na primeira configuração (Figura 13a), a aplicação e o software TIA Portal® encontram-se em computadores distintos. A troca de dados é feita a partir de documentos XML que podem ser importados/exportados a partir das aplicações desenvolvidas para cada computador. Os dados exportados do PC 2 podem ser modificados no PC 1 e reimportados

novamente para o PC 2. Por sua vez, na segunda configuração (Figura 13b), é utilizado apenas um computador no qual estão disponíveis o TIA Portal® e a aplicação desenvolvida.

No caso de uma situação típica de engenharia mecânica modular, a Siemens apresenta uma outra configuração baseada na de computador único. Suponhamos que o objetivo do conjunto TIA Portal-aplicação desenvolvida é programar várias máquinas semelhantes de variantes distintas. Para isto, deverá existir um projeto global no TIA Portal® que contenha os componentes de todas as variantes das máquinas. A aplicação desenvolvida deverá funcionar como gerador de código, ficando responsável por diversas tarefas, tais como: obtenção dos parâmetros para uma máquina específica e controlo da criação do projeto a partir da filtração e modificação dos elementos relevantes do projeto global (Siemens 2017c). A similaridade desta configuração com a situação em causa é evidente. No entanto, a aplicação desta configuração implica a criação de um projeto global que inclua todo o código que possa vir a ser utilizado. No caso específico da Consoveyo, S.A., iria implicar o desenvolvimento de um projeto que abrangesse todas as variantes possíveis, o que, dada a incerteza e elevado número das mesmas, seria uma tarefa extremamente exigente e complexa, e que exigiria, muito provavelmente, a alteração da forma como são desenvolvidos os projetos.

3.4.1.3 Aplicações Demo

A Siemens disponibiliza duas aplicações de demonstração em C#, a *StartOpenness* e o *DemoOpenness* (Siemens 2018). Estas ilustram algumas das funcionalidades passíveis de implementar e guiam o utilizador durante a programação da sua própria aplicação. Uma vez que o objetivo da presente dissertação é desenvolver uma aplicação que flexibilize a programação dos PLCs do *shuttle cars*, torna-se pertinente analisar estes exemplos.

A partir da análise do código e da observação da janela da aplicação *StartOpenness* (Figura 14) é possível identificar diversas funções, entre as quais:

- **Start TIA:** inicia o TIA Portal® com ou sem interface para o utilizador;
- **Dispose TIA:** mostra a instância do TIA Portal®;
- **Open Project:** abre um projeto existente;
- **Connect to open TIA Project:** conecta-se a uma instância do TIA Portal® ativa;
- **Save/Close Project:** grava/fecha o projeto aberto;
- **Add Device:** adiciona um novo hardware;
- **Compile:** faz a compilação do hardware adicionado.

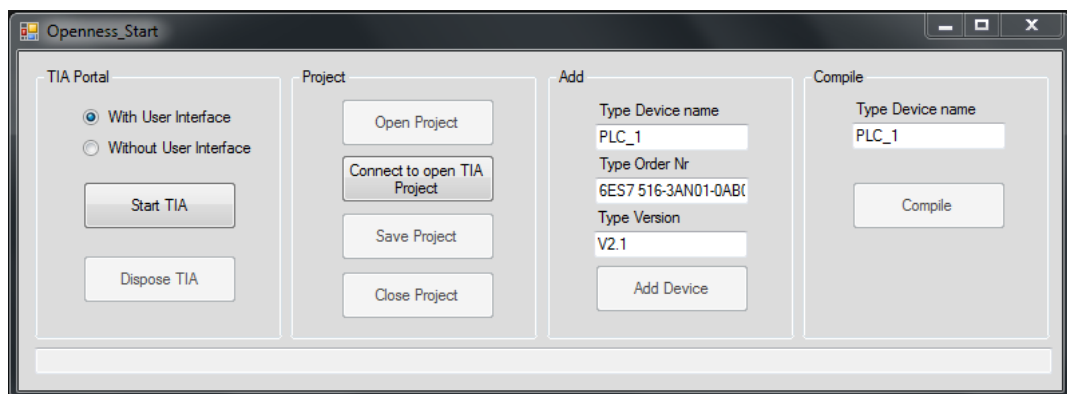


Figura 14 – Interface gráfica da aplicação *StartOpenness*.

Por sua vez, a aplicação *DemoOpenness* é composta por dois projetos, o *TIAOpennessHelper*, que contém métodos passíveis de serem usados numa aplicação própria, e o *TIAPortalOpennessDemo*, que pretende ilustrar o leque de funcionalidades que podem ser implementadas a partir desses métodos. Na Tabela 1 encontram-se listadas as principais funcionalidades e respetivos métodos. “TIA Portal Openness: Introduction and Demo Application” (Siemens 2016). Os métodos permitidos pela interface *Public API* encontram-se descritos em pormenor no manual “Automating projects with scripts” (Siemens 2017a).

Tabela 1 – Exemplos de funcionalidades e respetivos métodos implementáveis através do TIA Portal Openness®

Funcionalidades	Métodos
Iniciar TIA Portal	TiaPortal
Abrir/gravar/fechar projeto	Projects.Open(new FileInfo(filename)) Projects.Close() Projects.Save()
Conexão a instância do TIA Portal ativa	TiaPortal.GetProcesses() process.Attach()
Importar/exportar objetos	Import(file, importOption) Export(path, exportOption)
Apagar objetos	Delete()
Conexão online ao controlador	GoOnline() GoOffline()
Comparar controladores	CompareTo(Object)

3.4.1.4 Documentos XML e o TIA Portal

Dado que o TIA Portal Openness® permitia a exportação e importação de documentos XML com informação relativa a um projeto do TIA Portal®, avaliou-se a viabilidade da utilização dos mesmos na geração de código para PLCs, isto é, a possibilidade de montar e manipular um programa para PLCs partir dos seus documentos XML.

De modo a proceder à referida avaliação, recorreu-se a um simples teste manual descrito nos seguintes pontos:

1. Recorrendo à aplicação de demonstração *DemoOpenness* exportou-se um documento XML correspondente a um *program block* implementado no TIA Portal®;
2. Em seguida, editou-se manualmente o documento XML exportado de modo a que o *program block* apresentasse um código distinto;
3. Por último, procedeu-se à importação do documento XML manipulado, verificando-se que o código implementado no TIA Portal® tinha sido alterado com sucesso.

Uma vez provada a viabilidade desta abordagem, tornou-se essencial proceder a uma análise sobre os fundamentos, características e estrutura da linguagem XML, bem como sobre as formas de manipular documentos deste tipo através de uma aplicação de software em C#.

Definição de XML

O XML é uma linguagem *markup* desenvolvida pelo World Wide Web Consortium (W3C) que descreve uma classe de objetos denominados documentos XML (W3C 2016). Define um conjunto de regras para a codificação de documentos num formato legível, tanto pela máquina como pelo ser humano. Esta linguagem foi criada, originalmente, para enfrentar os desafios da publicação eletrónica em larga escala sendo que, atualmente, permite também o armazenamento e troca de informação de uma forma independente de qualquer software e hardware. Assim, um documento XML não transporta intrinsecamente nenhuma instrução acerca da apresentação dos dados, contendo exclusivamente informação (W3schools 2018a).

A linguagem XML foi desenvolvida com base em premissas específicas. Em primeiro lugar, esta linguagem deve ser facilmente utilizável pela internet, suportando uma ampla variedade de aplicações. Além disso, o desenvolvimento de programas para processamento de documentos XML deve ser relativamente elementar. Quanto ao próprio documento XML, este deve ser claro, fácil de criar e ter um *design* formal e conciso (W3C 2008).

Estrutura e Sintaxe do XML

Tal como se pode observar na Figura 15a), os documentos XML são construídos segundo uma estrutura em forma de árvore. O primeiro elemento da árvore – o elemento raiz (*root element*) – contém diversas ramificações – os elementos filhos (*child elements*). Cada elemento pode conter texto, atributos ou outros elementos. Na Figura 15b) pode observar-se a forma como são definidos os elementos raiz, os elementos filhos, os atributos e o texto (W3schools 2018c).

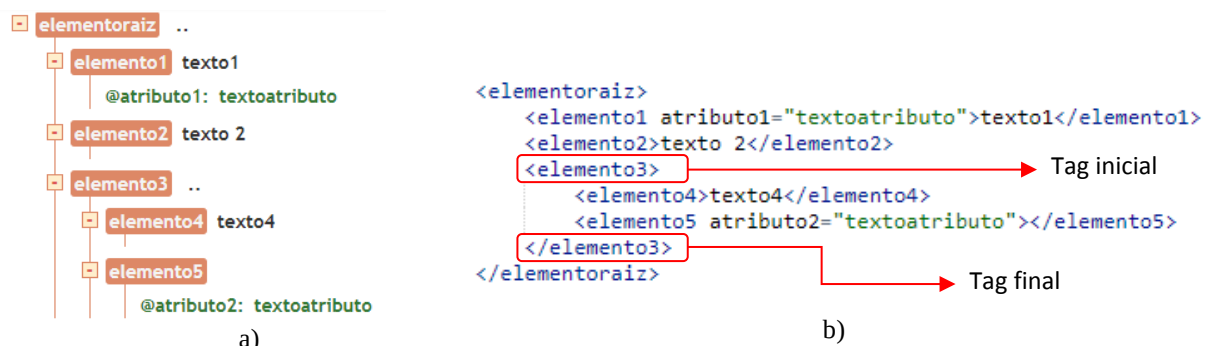


Figura 15 – Estrutura dos ficheiros XML: a) Estrutura em árvore; b) Exemplo de ficheiro XML.

A construção de documentos XML é feita segundo regras de sintaxe, listando-se em seguida algumas dessas diretivas (W3schools 2018b):

- Todos os elementos do XML têm uma *tag* inicial e uma *tag* final (Figura 15b);
- Geralmente, a primeira linha do ficheiro XML é composta por um prólogo (Figura 16). A sua inclusão é de carácter opcional e especifica a codificação usada na escrita do ficheiro. O prólogo não possui uma *tag* final dado que não faz parte do documento XML;

```
<?xml version="1.0" encoding="UTF-8"?>
```

Figura 16 – Prólogo do ficheiro XML

- As *tags* iniciais e finais dos elementos do XML são sensíveis a maiúsculas e minúsculas;

- Os comentários são escritos de acordo com a Figura 17.

```
<!-- This is a comment -->
```

Figura 17 – Comentários nos ficheiros XML.

A sintaxe de um documento XML pode ser validada a partir de um XML Schema (Figura 18). Os XML Schemas, igualmente escritos em linguagem XML, descrevem a estrutura à qual um determinado documento XML deve obedecer e contêm, por exemplo, os elementos e atributos do ficheiro e o tipo de dados associados aos mesmos.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xs:element name="produto">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nome" type="xsd:string" />
        <xs:element name="preço">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xsd:unsignedByte">
                <xs:attribute name="unidade" type="xsd:string" use="required" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="descrição">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="cor" type="xsd:string" />
              <xs:element name="tamanho" type="xsd:string" />
              <xs:element name="peso" type="xsd:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xsd:string" use="required" />
      <xs:attribute name="nomecategoria" type="xsd:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 18 – Exemplo de XML Schema.

Manipulação de ficheiros XML a partir de uma aplicação em C#

A leitura, escrita e edição de ficheiros XML a partir de um programa em C# pode ser realizada de formas distintas. Apresentam-se em seguida algumas das alternativas disponíveis para a manipulação de ficheiros deste tipo, ilustrando-se simples programas e respetivo resultado, no que diz respeito aos ficheiros XML produzidos e editados. Recomenda-se a leitura dos comentários esboçados nos troços de código para uma melhor compreensão dos programas.

XmlTextWriter e XmlTextReader (Figura 19): são classes que permitem ler e escrever ficheiros XML de acordo com as normas do W3C. Dado que não retêm o ficheiro em memória, possibilitam uma mais rápida leitura e escrita do ficheiro, não sendo, no entanto, possível fazer a sua edição.

```
// Construção do XML via XmlTextWriter
XmlTextWriter xmlWriter = new XmlTextWriter(
@"C:\Users\998720\Desktop\XmlTextWriter.xml", Encoding.UTF8);

// Escreve o "xmlTextWriter.xml"
xmlWriter.Formatting = Formatting.Indented; // Formata o ficheiro
xmlWriter.WriteStartElement("UP"); // <UP>
xmlWriter.WriteStartElement("FEUP"); // <FEUP>
xmlWriter.WriteAttributeString("Curso", "Mecânica"); // <FEUP curso = "Mecânica">
xmlWriter.WriteStartElement("Aluno"); // <Aluno>
xmlWriter.WriteString("Henrique"); // <Aluno>Henrique
xmlWriter.WriteEndElement(); // </Aluno>
xmlWriter.WriteEndElement(); // </FEUP>
xmlWriter.WriteEndElement(); // </UP>
xmlWriter.Close(); // Fecha o writer

// Leitura do XML via XmlTextReader
XmlTextReader xmlReader = new XmlTextReader(
@"C:\Users\998720\Desktop\XmlTextWriter.xml"); // Lê o ficheiro escrito
while (xmlReader.Read())
{
    if (xmlReader.Name == "Aluno") // No elemento <Aluno>
    {
        string Nome = xmlReader.ReadString(); // Lê o conteúdo
        Console.WriteLine(Nome);
    }
}
```



"xmlTextWriter.xml"

```
<UP>
  <FEUP Curso="Mecânica">
    <Aluno>Henrique</Aluno>
  </FEUP>
</UP>
```

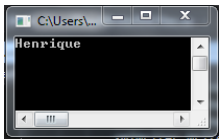


Figura 19 – Implementação do XmlTextReader e XmlTextWriter

XmlDocument (Figura 20): esta classe é uma representação em memória de um documento XML. Ao contrário do XmlTextReader, a classe XmlDocument permite ler, validar, editar e posicionar porções de XML. Através do XPath é possível navegar através dos elementos e atributos e estabelecer condições para alcançar um determinado nó ou conjunto de nós.

```
// Construção do XML via XmlDocument
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.Load(@"C:\Users\998720\Desktop\XmlDocument.xml"); // Leitura do ficheiro "xmlDocument.xml"
XmlNode nodeicbas = xmlDoc.CreateElement("ICBAS"); // <ICBAS></ICBAS>
XmlNode nodealuno = xmlDoc.CreateElement("Aluno"); // <Aluno></Aluno>
XmlAttribute attcurso = xmlDoc.CreateAttribute("Curso"); // Criação do atributo curso
attcurso.Value = "Medicina"; // Atribuição de um valor ao atributo
nodealuno.InnerText = "Manuel"; // Texto interior do node <Aluno>
nodeicbas.Attributes.Append(attcurso); // Atribuir atributo ao nó correspondente
nodeicbas.AppendChild(nodealuno); // Nó <Aluno> é filho do nó <ICBAS>
xmlDoc.DocumentElement.AppendChild(nodeicbas); // Nó <ICBAS> é filho do elemento raiz

xmlDoc.Save(@"C:\Users\998720\Desktop\XmlDocument.xml"); // Guarda ficheiro "xmlDocument.xml"

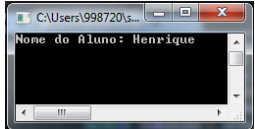
//Leitura do XML via XmlDocument
XmlDocument xmlDoc2 = new XmlDocument();
xmlDoc2.Load(@"C:\Users\998720\Desktop\XmlDocument.xml"); // Lê ficheiro "xmlDocument.xml"
XmlNode node1 = xmlDoc2.SelectSingleNode("UP/FEUP/Aluno"); // Seleciona nó <Aluno> dentro do nó <FEUP>
Console.WriteLine("Nome do Aluno" + node1.InnerText);

// Edição do XML via XmlDocument
XmlNode node1InnerXml = "HenriqueALTERADO"; // Altera conteúdo do "node1"
xmlDoc2.Save(@"C:\Users\998720\Desktop\XmlDocument_Alterado.xml");
```



"xmlDocument.xml"

```
<UP>
  <FEUP Curso="Mecânica">
    <Aluno>Henrique</Aluno>
  </FEUP>
  <ICBAS Curso="Medicina">
    <Aluno>Manuel</Aluno>
  </ICBAS>
</UP>
```





**"xmlDocument_A
lterado.xml"**

```
<UP>
  <FEUP Curso="Mecânica">
    <Aluno>HenriqueALTERADO</Aluno>
  </FEUP>
  <ICBAS Curso="Medicina">
    <Aluno>Manuel</Aluno>
  </ICBAS>
</UP>
```

Figura 20 – Implementação do XmlDocument.

XmlSerialize (Figura 21): Serializa e deserializa objetos para e a partir de documentos XML, respetivamente, permitindo controlar a forma como os objetos são codificados no XML. A serialização é o processo de conversão de objetos numa forma facilmente transportável, enquanto que a deserialização é o processo de reconstrução dos objetos. Os dados dos objetos são descritos a partir de construções de linguagem de programação como é o caso das classes. Estas podem ser criadas pelo próprio programador (Figuras 21 e 23b) ou geradas automaticamente a partir de um XML Schema (Figuras 22, 23a e 24).

```
// Construção do XML via XmlSerializer
try
{
    Product product = new Product // Novo produto
    {
        Id = "p01",
        NomeProduto = "Produto 1",
        NomeCategoria = "Categoria 1",
        Preço = new Price { Valor = 200, Unidade = "EUR" },
        Descrição = new Description { Cor = "Vermelho", Tamanho = "Tamanho 1", Peso = "100gr" }
    };
    /// Serialização: info presente nas classes preenche o XML
    XmlSerializer xmlSerialization = new XmlSerializer(typeof(Product));
    StreamWriter sw = new StreamWriter(
        @"C:\Users\998720\source\XmlSerialize.xml");
    xmlSerialization.Serialize(sw, product);
    sw.Close();
}
catch (Exception ex) { Console.WriteLine(ex.Message); }

// Leitura do XML via XmlSerializer
try
{
    // Deserialização: info do XML preenche as classes
    XmlSerializer xmlSerialization2 = new XmlSerializer(typeof(Product));
    StreamReader sr = new StreamReader(
        @"C:\Users\998720\source\XmlSerialize.xml");
    Product product = (Product)xmlSerialization2.Deserialize(sr);
    Console.WriteLine("Informação do produto: ");
    Console.WriteLine("Nome: " + product.NomeProduto);
    Console.WriteLine("Nome Categoria: " + product.NomeCategoria);
    Console.WriteLine("Id: " + product.Id);
    Console.WriteLine("Preço: " + product.Preço.Valor);
    Console.WriteLine("Unidade: " + product.Preço.Unidade);
    Console.WriteLine("Cor: " + product.Descrição.Cor);
    Console.WriteLine("Tamanho: " + product.Descrição.Tamanho);
    Console.WriteLine("Peso: " + product.Descrição.Peso);

    // Edição do XML via XmlSerializer
    product.NomeProduto = "Produto 1 Alterado";
    XmlSerializer xmlSerialization3 = new XmlSerializer(typeof(Product));
    StreamWriter sw3 = new StreamWriter(
        @"C:\Users\998720\Desktop\XmlSerialize_alterado.xml");
    xmlSerialization3.Serialize(sw3, product);
    sw3.Close();
}
catch (Exception ex) { Console.WriteLine(ex.Message); }
```

 "xmlSerialize.xml"

```
<produto xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
id="p01" nomecategoria="Categoria 1">
  <nome>Produto 1</nome>
  <preço unidade="EUR">200</preço>
  <descrição>
    <cor>Vermelho</cor>
    <tamanho>Tamanho 1</tamanho>
    <peso>100gr</peso>
  </descrição>
</produto>
```



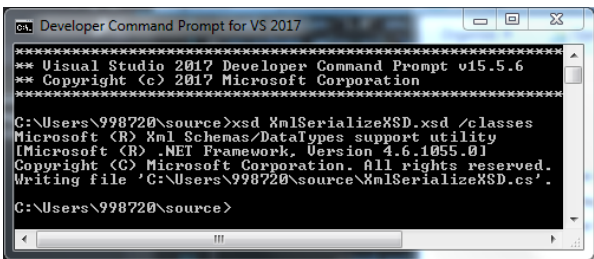
```
Informação do produto:
Nome: Produto 1
Nome Categoria: Categoria 1
Id: p01
Preço: 200
Unidade: EUR
Cor: Vermelho
Tamanho: Tamanho 1
Peso: 100gr
```

 "xmlSerialize_alterado.xml"

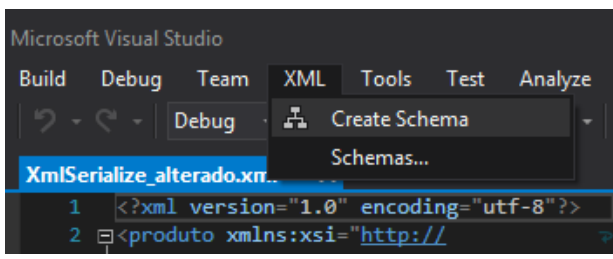
```
<produto xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
id="p01" nomecategoria="Categoria 1">
  <nome>Produto 1 Alterado</nome>
  <preço unidade="EUR">200</preço>
  <descrição>
    <cor>Vermelho</cor>
    <tamanho>Tamanho 1</tamanho>
    <peso>100gr</peso>
  </descrição>
</produto>
```

Figura 21 – Implementação do XmlSerialize a partir das classes manuais.

A obtenção das classes de forma automática pode ser realizada através do *Developer Command Prompt* (ferramenta do Microsoft Visual Studio®). Para isso, é necessário ter acesso ao XML Schema e utilizar o comando ilustrado na Figura 22a. Ao executar este comando é gerada uma classe (ficheiro .cs) com as classes necessárias para armazenar a informação dos objetos do documento XML. Caso não se disponha do XML Schema, é possível gerá-lo automaticamente através de um documento XML, utilizando o Visual Studio (Figura 22b).



a)



b)

Figura 22 – Obtenção das classes: a) Através do *Developer Command Prompt*; b) Geração do XML Schema.

```
// This source code was auto-generated by xsd, Version=4.6.1055.0.
[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType = true)]
[System.Xml.Serialization.XmlRootAttribute(Namespace = "", IsNullable = false)]
public partial class produto
{
    private string nomeField;
    private produtoPreço preçoField;
    private produtoDescrição descriçãoField;
    private string idField;
    private string nomecategoriaField;
    public string nome {
        get { return nomeField; }
        set { nomeField = value; }
    }
    public produtoPreço preço {
        get { return preçoField; }
        set { preçoField = value; }
    }
    public produtoDescrição descrição {
        get { return descriçãoField; }
        set { descriçãoField = value; }
    }
    public string id {
        get { return idField; }
        set { idField = value; }
    }
    public string nomecategoria {
        get { return nomecategoriaField; }
        set { nomecategoriaField = value; }
    }
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType = true)]
public partial class produtoPreço
{
    private string unidadeField;
    private byte valueField;
    public string unidade {
        get { return unidadeField; }
        set { unidadeField = value; }
    }
    public byte Value {
        get { return valueField; }
        set { valueField = value; }
    }
}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType = true)]
public partial class produtoDescrição
{
    private string corField;
    private string tamanhoField;
    private string pesoField;
    public string cor {
        get { return corField; }
        set { corField = value; }
    }
    public string tamanho {
        get { return tamanhoField; }
        set { tamanhoField = value; }
    }
    public string peso {
        get { return pesoField; }
        set { pesoField = value; }
    }
}
```

a)

```
// Classes
public class XmlSerialization
{
    [XmlRoot("produto")]
    public class Product
    {
        [XmlAttribute("id")]
        public string Id { get; set; }
        [XmlAttribute("nomecategoria")]
        public string NomeCategoria { get; set; }
        [XmlElement("nome")]
        public string NomeProduto { get; set; }
        [XmlElement("preço")]
        public Price Preço { get; set; }
        [XmlElement("descrição")]
        public Description Descrição { get; set; }
    }

    public class Price
    {
        [XmlAttribute("unidade")]
        public string Unidade { get; set; }
        [XmlText]
        public int Valor { get; set; }
    }

    public class Description
    {
        [XmlElement("cor")]
        public string Cor { get; set; }
        [XmlElement("tamanho")]
        public string Tamanho { get; set; }
        [XmlElement("peso")]
        public string Peso { get; set; }
    }
}
```

b)

Figura 23 - Classes obtidas: a) Automaticamente a partir do Schema; b) Manualmente pelo programador

```
// Construção do XML via XmlSerializer com classes automáticas
try
{
    produto product = new produto // Novo produto
    {
        id = "pa01",
        nome = "Auto 1",
        nomecategoria = "Categoria 1",
        preço = new produtoPreço { Value = 200, unidade = "EUR" },
        descrição = new produtoDescrição { cor = "Vermelho", tamanho = "Tamanho 1", peso = "100gr" }
    };

    //// Serialização: info presente nas classes preenche o XML
    XmlSerializer xmlSerialAuto = new XmlSerializer(typeof(produto));
    StreamWriter swAuto = new StreamWriter(
        @"C:\Users\998720\source\XmlSerialAuto.xml");
    xmlSerialAuto.Serialize(swAuto, product);
    swAuto.Close();
}
catch (Exception ex) { Console.WriteLine(ex.Message); }

// Leitura do XML via XmlSerializer com classes automáticas
try
{
    // Deserialização: info do XML preenche as classes
    XmlSerializer xmlSerialAuto2 = new XmlSerializer(typeof(Produto));
    StreamReader srAuto = new StreamReader(
        @"C:\Users\998720\source\XmlSerialAuto.xml");
    Produto productAuto = (Produto)xmlSerialAuto2.Deserialize(srAuto);

    // Edição do XML via XmlSerializer
    productAuto.NomeProduto = "Auto 1 Alterado";
    XmlSerializer xmlSerialAuto3 = new XmlSerializer(typeof(Produto));
    StreamWriter swAuto3 = new StreamWriter(
        @"C:\Users\998720\Desktop\XmlSerialAuto_Alterado.xml");
    xmlSerialAuto3.Serialize(swAuto3, productAuto);
    swAuto3.Close();
}
catch (Exception ex) { Console.WriteLine(ex.Message); }
```

```
<produto xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
" id="pa01" nomecategoria="Categoria 1">
  <nome>Auto 1</nome>
  <preço unidade="EUR">200</preço>
  <descrição>
    <cor>Vermelho</cor>
    <tamanho>Tamanho 1</tamanho>
    <peso>100gr</peso>
  </descrição>
</produto>
```

```
<produto xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
" id="p01" nomecategoria="Categoria 1">
  <nome>Produto 1 Alterado</nome>
  <preço unidade="EUR">200</preço>
  <descrição>
    <cor>Vermelho</cor>
    <tamanho>Tamanho 1</tamanho>
    <peso>100gr</peso>
  </descrição>
</produto>
```

Figura 24 – Implementação do XmlSerialize a partir das classes automáticas.

Como se pode constatar pela comparação das Figuras 21 e 24, os ficheiros XML gerados e manipulados a partir das classes criadas manualmente e das geradas automaticamente possuem uma estrutura idêntica. No entanto, em alguns casos, as classes geradas automaticamente contêm erros, tornando impossível a sua utilização.

3.4.2 Sharp 7

Tal como referido no Capítulo 2, o Sharp 7 possibilita uma série de funções através da sua interface para o Microsoft Visual Studio®. A Tabela 2 apresenta um resumo das funções passíveis de implementar utilizando o Sharp 7 e da compatibilidade das funções desta ferramenta com as diversas séries de PLCs da Siemens.

Tabela 2 – Funções disponíveis no Sharp7

Funções	CPU				Descrição da função
	300	400	1200	1500	
<i>ConnectTo</i>	X	X	X	X	Conexão ao PLC
<i>Read/Write DBs</i>	X	X	X	X	Lê (Escreve) parte de uma DB do (para o) PLC.
<i>Read/Write EBs</i>	X	X	X	X	Lê (Escreve) parte de uma IPU ⁽¹⁾ do (para o) PLC.
<i>Read/Write ABs</i>	X	X	X	X	Lê (Escreve) parte de uma IPI ⁽¹⁾ do (para o) PLC.
<i>Read/Write MKs</i>	X	X	-	-	Lê (Escreve) parte de uma <i>Merkers area</i> do (para o) PLC.
<i>Read/Write TMs</i>	X	X	-	-	Lê (Escreve) temporizadores do (para o) PLC.
<i>Read/Write CTs</i>	X	X	-	-	Lê (Escreve) contadores do (para o) PLC.
<i>Get/Set PLC Date/Time</i>	X	X	-	-	Obtém/Estabelece a data e hora do PLC
<i>PLC Status</i>	X	X	-	-	Obtém o estado do PLC
<i>Get AG Block Info</i>	X	X	-	-	Devolve informação de um bloco (Tipo, nº, etc.)
<i>DB Get/Fill</i>	X	X	-	-	Lê/Escreve uma DB completa sem ser necessário especificar o seu tamanho
<i>Control Run/Stop</i>	X	X	-	-	Possibilita colocar o CPU em Run/Stop
<i>System info functions</i>					Fornece informação sobre o CPU (order code, etc.)
<i>Security functions</i>	X	X	-	-	Fornece informação acerca do nível de proteção e possibilita estabelecer passwords

(1) – IPU: Inputs Process Image; IPI: Outputs Process Image. São áreas de memória para as (ou das) quais são copiados os *inputs/outputs* físicos.

Com vista a uma melhor compreensão das potencialidades desta ferramenta, foi desenvolvido um programa em TIA Portal® (Figura 25), contendo apenas um *organization block* (com um contacto e uma bobina - Figura 25b) e uma *global database blocks* (com vários tipos de variáveis - Figura 25c). Com base neste programa foi desenvolvida uma aplicação de consola em C# no Microsoft Visual Studio® para implementar algumas das funções descritas na Tabela 2.

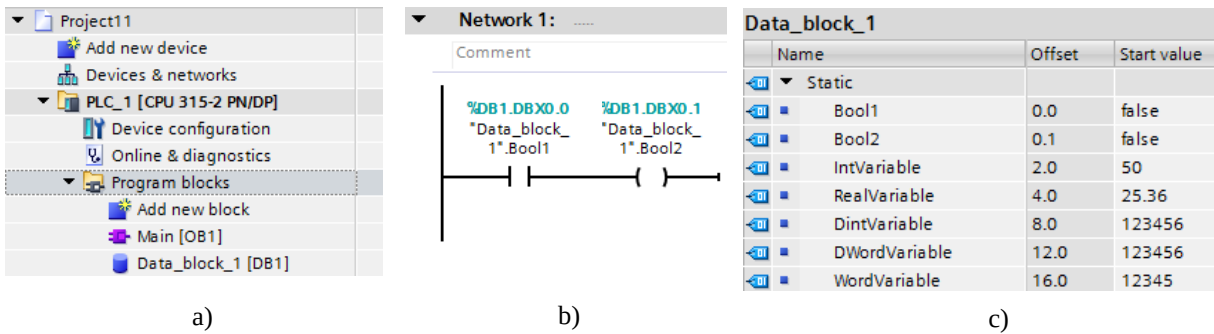


Figura 25 – Programa em TIA Portal para teste do Sharp 7: a) *TreeView*; b) *Organization block*; c) *Global database block*.

Em primeiro lugar, dado que o Sharp 7 comunica via TCP/IP, foi necessário criar um servidor, usando o NetToPLCsim (Figura 26), que permitisse aceder ao simulador do PLC do TIA Portal® por esta via.

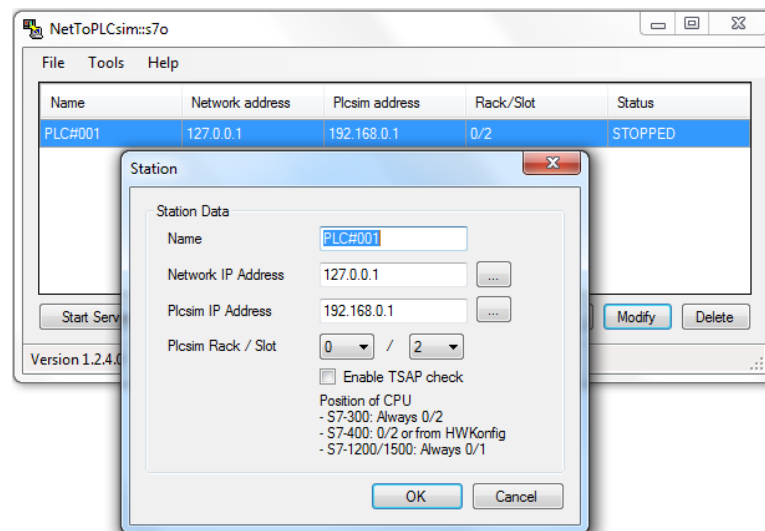


Figura 26 – Criação do servidor para conexão ao PLC a partir do NetToPLCsim.

Seguidamente, implementaram-se algumas das funções descritas na Tabela 2. Recomenda-se a leitura dos comentários presentes nos troços de código para uma melhor compreensão das funções implementadas. Na Figura 27 apresenta-se a função ConnectTo() que permite fazer a conexão ao PLC que se encontra online.

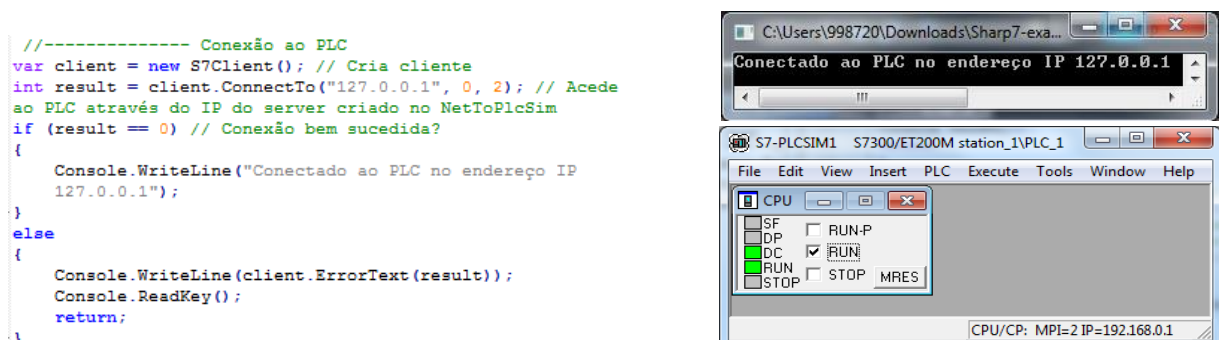


Figura 27 – Implementação da função ConnectTo().

Na Figura 28 apresenta-se a implementação função DBRead(). Como se pode constatar, as variáveis lidas e dispostas na consola são aquelas que se encontram na DB *Data_block_1*, presente no PLC.

```

//----- Lê os valores da DB1
Console.WriteLine("\n----- Leitura DB1");
byte[] db1Buffer = new byte[18]; // Length = n° de bytes usados na DB
// Contém valores das variáveis da DB
result = client.DBRead(1, 0, db1Buffer.Length, db1Buffer);
if (result != 0)
{
    Console.WriteLine("Error: " + client.ErrorText(result));
}
bool db1dbx00 = S7.GetBitAt(db1Buffer, 0, 0); // Lê Bool1
Console.WriteLine("Bool1 | DB1.DBX0.0: " + db1dbx00);
bool db1dbx01 = S7.GetBitAt(db1Buffer, 0, 1); // Lê Bool2
Console.WriteLine("Bool2 | DB1.DBX0.1: " + db1dbx01);
int db1dbw2 = S7.GetIntAt(db1Buffer, 2); // Lê IntVariable
Console.WriteLine("IntVariable | DB1.DBW2: " + db1dbw2);
double db1dbd4 = S7.GetRealAt(db1Buffer, 4); // Lê RealVariable
Console.WriteLine("RealVariable | DB1.DBD4: " + db1dbd4);
double db1dbd8 = S7.GetDintAt(db1Buffer, 8); // Lê DIntVariable
Console.WriteLine("DIntVariable | DB1.DBD8: " + db1dbd8);
uint db1dbd12 = S7.GetDwordAt(db1Buffer, 12); // Lê DWordVariable
Console.WriteLine("DWordVariable | DB1.DBD12: " + db1dbd12);
ushort db1dbw16 = S7.GetWordAt(db1Buffer, 16); // Lê WordVariable
Console.WriteLine("WordVariable | DB1.DBD16: " + db1dbw16);
string hexdb1dbd12 = db1dbd12.ToString("X"); // Valor em Hexadecimal
Console.WriteLine("DWordVariable(Hex) | DB1.DBD12: " + hexdb1dbd12);
string hexdb1dbw16 = db1dbw16.ToString("X"); // Valor em Hexadecimal
Console.WriteLine("WordVariable(Hex) | DB1.DBD16: " + hexdb1dbw16);

```

Name	Start value	Monitor value
Static		
Bool1	false	FALSE
Bool2	false	FALSE
IntVariable	50	50
RealVariable	25.36	25.36
DintVariable	123456	123456
DWordVariable	123456	16#0001_E240
WordVariable	12345	16#3039

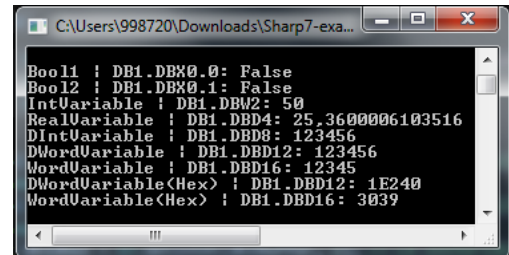


Figura 28 – Implementação da função DBRead().

De salientar que a escrita de novos valores no PLC para as variáveis da *Data_block_1* pode ser feita com recurso à função *s7MultiVar.Write()* (Figura 29).

```

var s7MultiVar_ = new S7MultiVar(client);
byte[] db1_ = new byte[12]; // N° de bytes a modificar
S7.SetBitAt(ref db1_, 0, 0, true); //Address 0.0 => true
S7.SetIntAt(db1_, 2, 15); //Address 2.0 => 15
S7.SetRealAt(db1_, 4, (float)69.9); //Address 4.0 => 69.9
S7.SetDintAt(db1_, 8, 234213); //Address 8.0 => 234213
s7MultiVar_.Add(S7Consts.S7AreaDB, S7Consts.S7WLBByte, 1, 0, db1_.Length, ref db1_); // Escrita de multiplas variáveis
result = s7MultiVar_.Write();
if (result != 0)
{
    Console.WriteLine("Error on s7MultiVar.Read()");
}

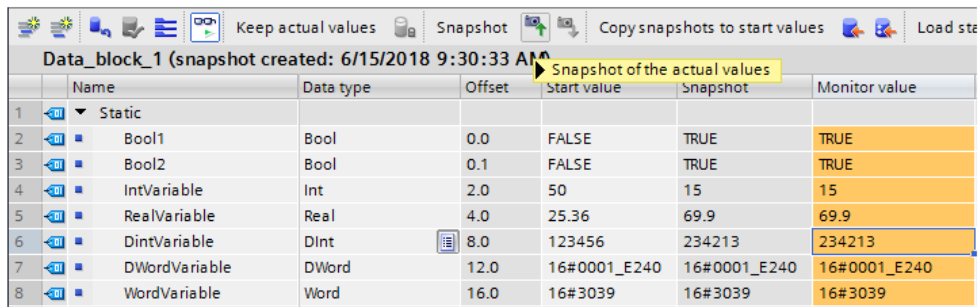
```

Name	Start value	Monitor value
Static		
Bool1	false	TRUE
Bool2	false	TRUE
IntVariable	50	15
RealVariable	25.36	69.9
DintVariable	123456	234213
DWordVariable	123456	16#0001_E240
WordVariable	12345	16#3039

Figura 29 – Implementação da função *s7MultiVar_.Write()*.

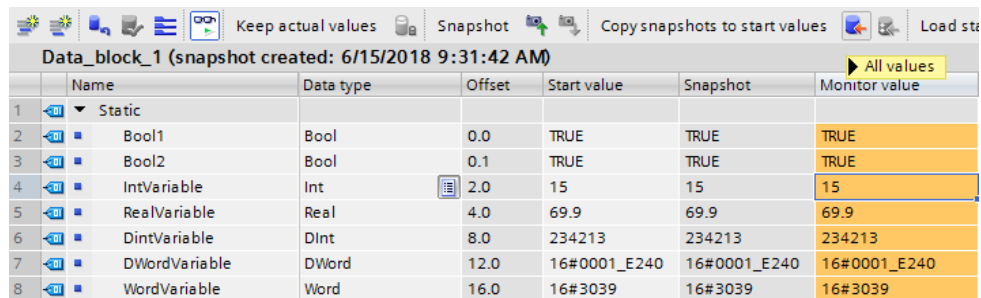
Note-se que, no entanto, a escrita de variáveis na *Data_block_1* é apenas feita no PLC, não alterando os valores das variáveis da DB existente no programa do TIA Portal®. Assim, ao reiniciar o PLC, os valores iniciais da DB mantêm-se inalterados. De modo a que a alteração seja definitiva, é necessário que estes valores sejam transferidos para os valores iniciais da DB. Esta transferência é possível a partir das funções *Snapshot* e *Copy snapshots to start values* disponíveis no TIA Portal®. A primeira, ilustrada na Figura 30, permite capturar os valores das variáveis atuais, enquanto que, a segunda (Figura 31) permite transferir esses valores para os valores iniciais da DB.

Dado que é possível alterar as DBs existentes no programa do TIA Portal®, o Sharp 7 poderá ser utilizado na flexibilização da programação de PLCs. Através da alteração de variáveis poderá configurar-se um dado programa de PLC de modo a que este replique o comportamento desejado.



Data_block_1 (snapshot created: 6/15/2018 9:30:33 AM)						
	Name	Data type	Offset	Start value	Snapshot	Monitor value
1	Static					
2	Bool1	Bool	0.0	FALSE	TRUE	TRUE
3	Bool2	Bool	0.1	FALSE	TRUE	TRUE
4	IntVariable	Int	2.0	50	15	15
5	RealVariable	Real	4.0	25.36	69.9	69.9
6	DintVariable	Dint	8.0	123456	234213	234213
7	DWordVariable	DWord	12.0	16#0001_E240	16#0001_E240	16#0001_E240
8	WordVariable	Word	16.0	16#3039	16#3039	16#3039

Figura 30 – Função de captura dos valores atuais.



Data_block_1 (snapshot created: 6/15/2018 9:31:42 AM)						
	Name	Data type	Offset	Start value	Snapshot	Monitor value
1	Static					
2	Bool1	Bool	0.0	TRUE	TRUE	TRUE
3	Bool2	Bool	0.1	TRUE	TRUE	TRUE
4	IntVariable	Int	2.0	15	15	15
5	RealVariable	Real	4.0	69.9	69.9	69.9
6	DintVariable	Dint	8.0	234213	234213	234213
7	DWordVariable	DWord	12.0	16#0001_E240	16#0001_E240	16#0001_E240
8	WordVariable	Word	16.0	16#3039	16#3039	16#3039

Figura 31 – Transferência dos valores atuais para os valores iniciais.

3.5 Abordagem selecionada

Uma vez exploradas as ferramentas, constatou-se que a utilização do TIA Portal Openness® permitia implementar um vasto leque de funcionalidades, entre as quais funções de interface com o TIA Portal® e de exportação/importação de documentos XML com informação relativa ao hardware e software presentes num programa. Com um simples teste provou-se que era possível alterar o código presente num programa do TIA Portal® através dos seus documentos XML. Neste sentido, identificaram-se métodos que possibilitavam a manipulação de documentos XML a partir de uma aplicação de software externa. Tal facto sugeriu que seria possível desenvolver uma aplicação externa para geração de código que manipulasse documentos XML perante uma dada configuração do sistema a controlar. Por outro lado, o Sharp 7 permitia apenas comunicar com o PLC e alterar valores de variáveis em diversas áreas de um programa PLC. A utilização desta ferramenta implicaria que, independentemente da configuração desejada, o código presente no programa manter-se-ia inalterado. A parametrização das variáveis permitiria, posteriormente, e mediante a configuração desejada, seleccionar o código a ser executado, resultando num programa mais pesado e difícil de manter. Assim, concluiu-se que o TIA Portal Openness® revelava um maior potencial para a flexibilização da programação de PLCs.

Posto isto, e perante as vantagens inerentes à geração de código, optou-se pela utilização do TIA Portal Openness®, direccionando assim a presente dissertação para esta abordagem.

No entanto, uma vez utilizado o TIA Portal Openness®, não foi possível desenvolver uma solução que incorporasse o programa de controlo atualmente em uso nos *shuttle cars* da Consoveyo, S.A. dado que este foi desenvolvido no SIMATIC STEP7®. Além disso, tal como exposto na Secção 2.4.1.1, o desenvolvimento de uma ferramenta com a capacidade de gerar e manipular código para PLC requer total entendimento do programa e do sistema a controlar, o que seria uma tarefa demasiado complexa e extensa perante o curto prazo da dissertação.

Assim, optou-se por desenvolver um novo programa de controlo em diagrama de contactos, utilizando o TIA Portal®, dado que este permitia utilizar a interface do TIA Portal

Openness®. Além disso, selecionou-se o FactoryIO® para simular o comportamento do sistema perante o programa desenvolvido.

O novo programa de controlo foi desenvolvido tentando replicar ao máximo os princípios de funcionamento e as variações implementadas nos *shuttle cars* da Consoveyo, S.A., referidos no Capítulo 3.2.

No entanto, devido ao curto prazo da dissertação e a diversas limitações impostas pelo simulador utilizado, o programa desenvolvido e, consequentemente, as variações a considerar na implementação do *shuttle car* ficaram condicionados. Assim, consideraram-se algumas das variações implementadas nos *shuttle cars* da Consoveyo, S.A. (Capítulo 3.2) nomeadamente:

- o número de linhas de carga e descarga do *shuttle car* e o seu posicionamento;
- o número de *conveyors* por linha;
- velocidades do *shuttle car* (apenas em modo manual).

Além disso, adotaram-se outras variações que se mostraram relevantes durante o desenvolvimento do programa, entre as quais:

- inclusão de opcionais (sinalizadores indicativos da disponibilidade das linhas – Capítulo 4.1);
- alteração de limites de software e posição de standby;
- outros parâmetros (tempos de carga e descarga e tempo de transição entre *conveyors*).

Quanto à aplicação de geração de código optou-se pelo seu desenvolvimento através da manipulação de ficheiros XML em C#, dado ser uma linguagem de fácil aprendizagem e por permitir a utilização da interface do TIA Portal Openness®.

4 Programa de controlo e ambiente de simulação desenvolvidos

O desenvolvimento do novo programa de controlo do *shuttle car*, referido na Secção 3.5, dependia inteiramente do sistema a controlar e, conseqüentemente, do ambiente de simulação a construir.

Este capítulo foca-se, assim, na definição do sistema a controlar, bem como na descrição do comportamento obtido através da implementação do programa desenvolvido no sistema em estudo. Por último, o programa de controlo desenvolvido em TIA Portal® será ainda apresentado e extensamente analisado.

4.1 Ambiente de simulação

O ambiente de simulação em FactoryIO® foi desenvolvido não só com o intuito de definir o sistema a controlar, mas também de testar o programa realizado no TIA Portal®. Para além disso, a construção de um ambiente de simulação facilita a visualização do comportamento do sistema e da aplicação de geração de código desenvolvida.

Ao optar pelo desenvolvimento de um novo programa de controlo e pela utilização do FactoryIO® para a construção dos ambientes de simulação, foi necessário redefinir ligeiramente o sistema a controlar. Isto porque este simulador, devido ao seu carácter educativo e às suas limitações, não permite implementar todos os constituintes presentes num armazém automático da Consoveyo, S.A, nem replicar o seu funcionamento. Assim, o programa de controlo desenvolvido teve de ser adaptado a estas restrições.

4.1.1 Sistema a controlar

Para a definição do sistema a controlar foi desenvolvido um *layout*-tipo no FactoryIO® à semelhança dos sistemas produzidos pela Consoveyo, S.A., o qual é descrito seguidamente.

Como se pode constatar pela observação das Figuras 32a e 33, o *layout*-tipo é constituído por um transelevador de garfos. O princípio de funcionamento destes equipamentos é muito semelhante ao do *shuttle car*, à exceção das direções segundo as quais é permitido fazer o posicionamento: os transelevadores movimentam-se segundo as direções horizontal e vertical, enquanto que os *shuttle cars* se posicionam apenas segundo a direção horizontal. Assim, adaptou-se o funcionamento do transelevador para que este assumisse o comportamento do *shuttle car*. Atente-se a que, apesar do transelevador “adaptado” permitir posicionamento apenas segundo a direção horizontal, é necessária uma ligeira movimentação vertical aquando da carga e descarga de paletes.

Além disso, o *layout* engloba também diversas linhas de carga e descarga, compostas por diferentes números de *conveyors*, dispostas ao longo do curso do *shuttle car* (Figuras 32a e 33). A carga, transportada em paletes, é introduzida no sistema a partir de *emitters* (identificados na Figura 32a por setas verdes) e extraída por *removers* (identificados na Figura 32a por setas vermelhas). As linhas de carga são responsáveis pela receção e encaminhamento das paletes desde a entrada da linha até ao ponto de carga. Por sua vez, as linhas de descarga recebem as paletes descarregadas pelo *shuttle* e encaminham-nas até ao seu destino final.

Adicionalmente, de modo a possibilitar o controlo do sistema instalaram-se diversos elementos sensores: nos *conveyors* de entrada de cada linha (dois) e nos restantes (apenas um) (Figuras 32a e 34), para possibilitar o controlo dos mesmos; na ala do *shuttle car* (um em cada extremo) (Figuras 33 e 34), para implementação da redução de velocidade do *shuttle car*.

Por último, e com o objetivo de indicar a disponibilidade de cada linha para receber carga, incorporam-se sinalizadores luminosos, do tipo coluna, em várias linhas de *conveyors* (Figura 32b).

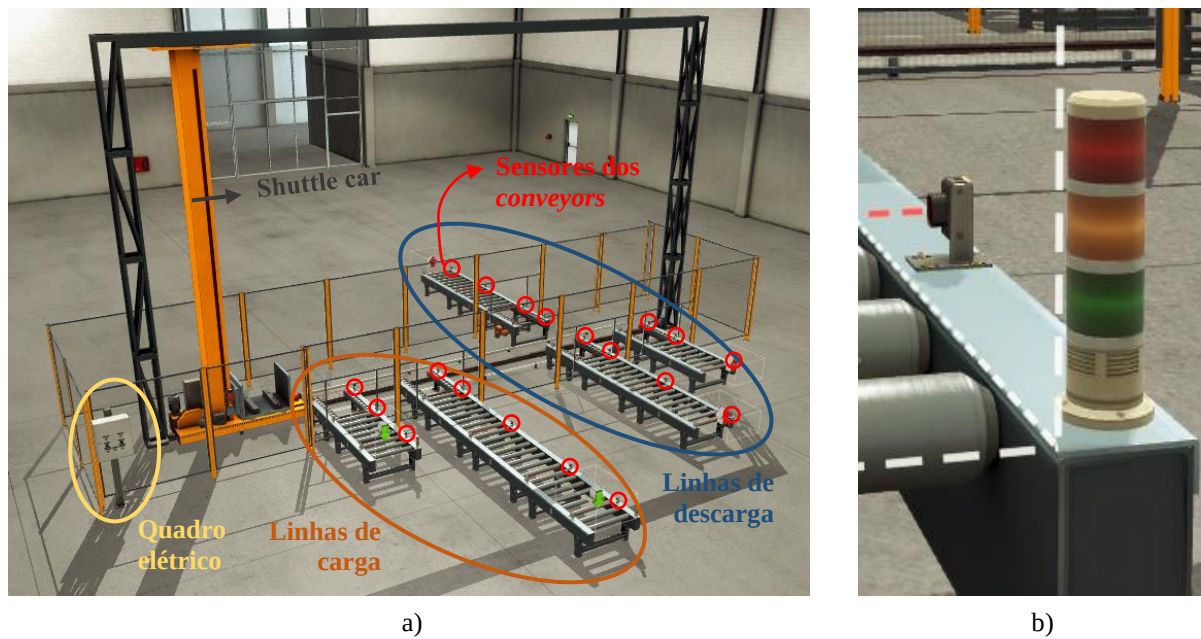


Figura 32 – *Layout* desenvolvido no FactoryIO: a) Vista lateral; b) Pormenor dos sensores dos conveyors e sinalizadores indicativos da disponibilidade da linha.

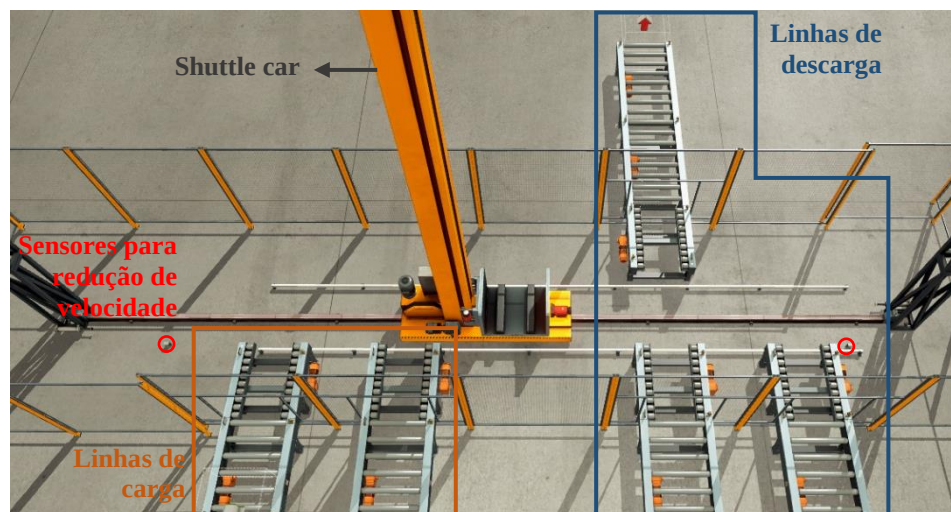


Figura 33 - *Layout* desenvolvido no FactoryIO (Vista superior).

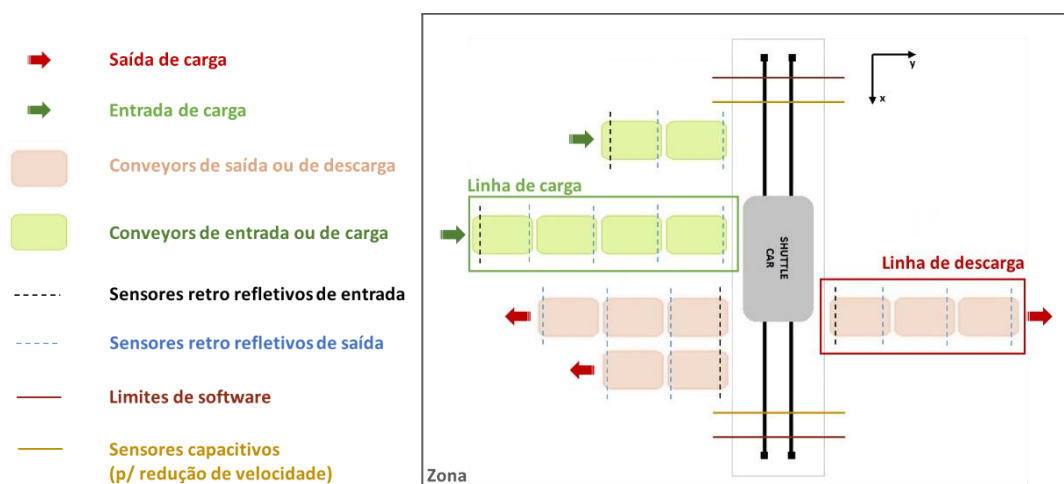


Figura 34 – Esquematisação do *layout* desenvolvido.

4.1.2 Funcionamento do sistema controlado

Tal como no caso do sistema desenvolvido pela Consoveyo, S.A., o sistema implementado possui dois modos de funcionamento, os quais são descritos seguidamente.

Modo Manual

A movimentação do *shuttle car* em modo manual é conseguida através do quadro elétrico presente no *layout* (Figuras 32a e 35). Para ativar este modo é necessário premir a botoneira “Reset”, seleccionando posteriormente o modo “Manual” a partir do seletor “Auto-Manual”. A seleção da velocidade é efetuada através das botoneiras “Movimento +” e “Movimento –”. Premindo, respetivamente, cada uma destas botoneiras a velocidade é incrementada e decrementada até três vezes. Quando as velocidades máximas positivas ou negativas são seleccionadas, os sinalizadores das botoneiras “Movimento +” e “Movimento –” são, respetivamente, desligados. Por sua vez, quando a velocidade é nula, o sinalizador da botoneira “Translação” é desligado.

Neste modo é ainda possível ter dois comportamentos distintos: com o seletor “0-ByPass” na posição “0” é apenas permitido movimentar o *shuttle car* dentro dos limites de software; com o seletor na posição “By-Pass” é possível ultrapassar esses limites.



Figura 35 – Painel de controlo do quadro elétrico.

Modo Automático

No modo automático as movimentações do *shuttle car* e dos *conveyors* são controladas automaticamente a partir do programa de controlo.

O *shuttle car* recebe uma nova ordem de transporte sempre que se encontra pronto, ou seja, quando está posicionado entre os limites de software (Figura 34) e não tem nenhuma operação de carga ou descarga em curso. As ordens são cumpridas por ordem de chegada, não tendo sido implementado nenhum sistema de otimização.

A cada nova ordem de transporte o programa verifica se há condições para que esta seja cumprida. Assim, é verificado: se existe uma carga a transportar, a partir do sensor existente no final da linha correspondente à coordenada de origem indicada na ordem recebida; se o destino se encontra livre, a partir do sensor colocado no início da linha correspondente à coordenada de destino da ordem recebida. Caso não se reúnam condições para que a ordem recebida seja executada, o transporte não é efetuado e o *shuttle car* posiciona-se na posição de *standby*, após alguns segundos.

4.2 Programa de Controlo do *shuttle car*

Tal como referido na Secção 3.5 optou-se pelo desenvolvimento de um novo programa em TIA Portal® para o controlo do *shuttle car*. O programa foi desenvolvido tendo em conta as considerações expostas na Secção 2.4.1.1 acerca da modularidade necessária à geração de código e englobando as variantes identificadas na Secção 3.5, de modo a que pudesse ser aplicado em múltiplas configurações do sistema descrito na Secção 4.1.1.

4.2.1 Requisitos de Modularidade

A construção de um programa modular apresenta dois benefícios principais: a simplificação de um sistema complexo em subsistemas mais simples; e a reutilização de módulos de código.

O desenvolvimento de soluções deste tipo obriga a alterações no método de trabalho, uma vez que o objetivo é encontrar uma solução para um tipo de problema, e não para um problema em particular.

Assim, de modo a tipificar o problema com o objetivo de criar diversos módulos de código modulares, dividiu-se o sistema descrito no ponto 4.1.1 em vários subsistemas distintos. Descreve-se, em seguida, o raciocínio envolvido na conceção dos vários módulos a implementar no programa (Figura 36):

- O programa desenvolvido deveria conter dois módulos de código, para cada um dos modos de funcionamento do *shuttle car* (automático e manual);
- Deveria existir um módulo de código que permitisse a leitura das ordens de transporte a partir de uma listagem. A ordem lida deveria ser transmitida ao módulo de funcionamento automático do *shuttle car*;
- Deveriam existir módulos de código que avaliassem a disponibilidade das linhas de descarga para receber carga, bem como a presença de cargas a transportar nas linhas de carga, de modo a controlar a execução das ordens. Esta informação deveria ser transmitida ao módulo de funcionamento automático do *shuttle car*;
- Deveriam ser criados três módulos de código que permitissem o controlo dos primeiros *conveyors*, dos *conveyors* intermédios, e dos últimos *conveyors* de cada linha. Deste modo, o controlo de todas as linhas de carga e descarga seria implementado com recurso aos mesmos três módulos.

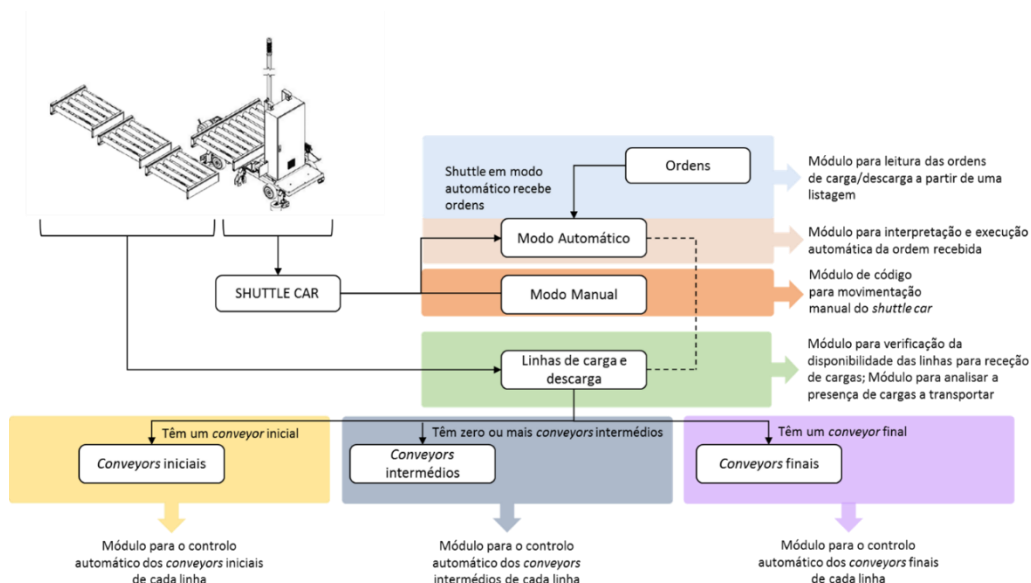


Figura 36 – Esquematização da modularização do sistema em estudo.

4.2.2 Componentes do programa desenvolvido em TIA Portal®

Uma vez concebidos os módulos necessários ao controlo do sistema, foi necessário proceder à sua implementação no TIA Portal®, através de *function blocks* (FBs). Isto porque, como referido na Secção 2.2.2.2, a utilização de *function blocks* (FBs) permite reutilizar a mesma lógica em sistemas distintos, através de diferentes instanciações.

Para além disso, no decorrer do desenvolvimento do programa, foi necessário criar outros elementos de software entre os quais *organization blocks* (OBs), *global database blocks* (DBs) e *tagtables*.

Seguidamente é feita uma breve descrição da totalidade dos elementos de software incorporados no programa desenvolvido no TIA Portal®.

Function Blocks

Como foi referido anteriormente os *function blocks* concebidos no TIA Portal® correspondem aos módulos desenvolvidos na Secção 4.2.1. Em seguida, são exploradas as funções implementadas por cada um deles, ilustrando-se alguns GRAFCETS concebidos com vista à sua programação. Os restantes GRAFCETS podem ser encontrados no Anexo B.

FB Reading_List: Lê a ordem de índice *i*, a partir da lista de ordens *Orders* (descrita posteriormente na secção Database Blocks), extraíndo as coordenadas de carga (X_Load e Y_Load) e de descarga (X_Unload e Y_Unload). Este FB tem a particularidade de ser programado em texto estruturado. Isto porque, tal como se pode constatar pela observação da Figura 37, com o PLC utilizado, não é possível fazer a indexação de um vetor através de uma variável inteira em diagrama de contactos eléctricos.

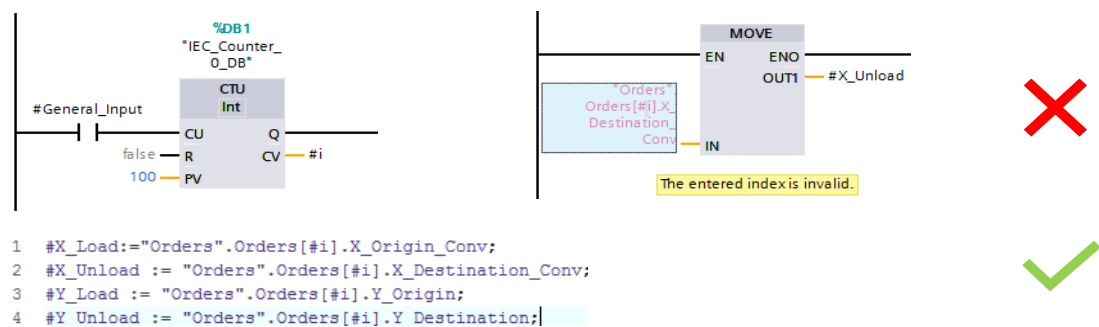


Figura 37 – Indexação de um vetor através de uma variável inteira.

FB Conv 1: Cada instância deste *function block*, cujo GRAFCET é visível na Figura 38, permite controlar o primeiro *conveyor* de uma linha de carga ou descarga do *shuttle car*.

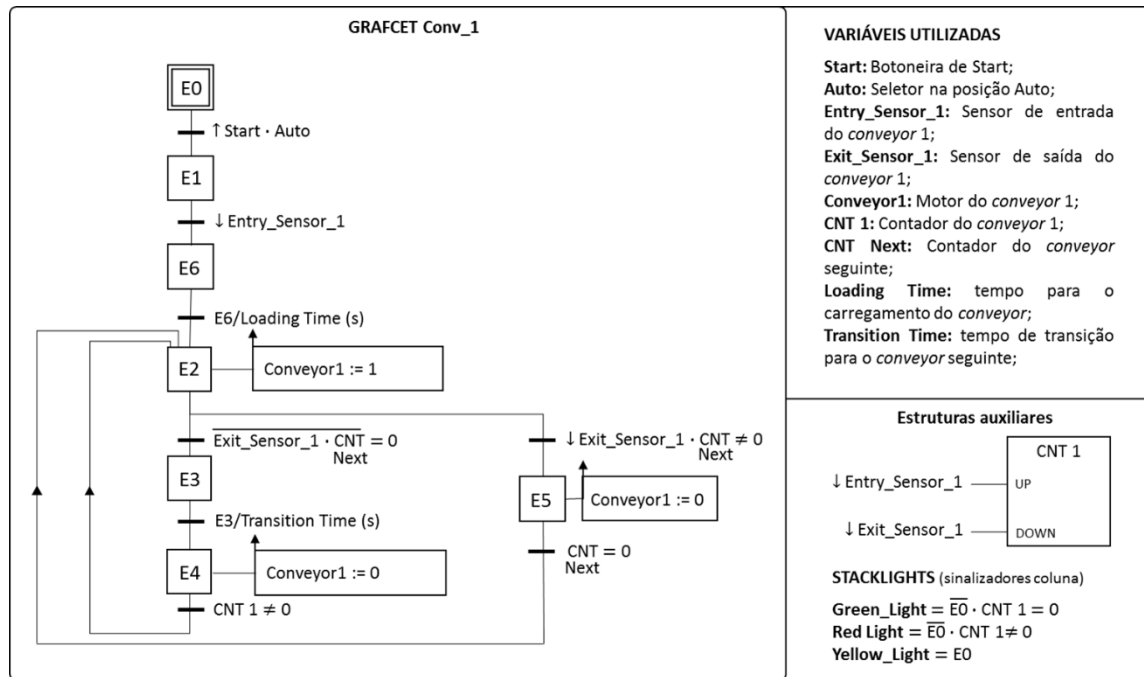


Figura 38 – GRAFCET do *function block* Conv_1.

FB Conv n: Cada instância deste *function block* permite controlar o último *conveyor* de uma linha de carga ou descarga do *shuttle car*.

FB Conv i: Cada instância deste *function block* permite controlar um dos *conveyors* intermédios de uma linha de carga ou descarga do *shuttle car*. Assim, para uma dada linha este bloco é instanciado tantas vezes quanto o número de *conveyors* intermédios da linha. Os *conveyors* intermédios de uma dada linha são aqueles que se encontram posicionados entre o primeiro e último *conveyors*.

FB Reading Array: Garante o cumprimento das ordens de forma sequencial. Permite a leitura da ordem seguinte (a partir do *function block* Reading_List) quando o *shuttle car* está pronto, ou seja, quando está posicionado entre os limites de software (Figura 34) e não tem nenhuma operação de transporte em curso. As coordenadas de carga e descarga presentes na ordem lida são passadas ao *function block* que controla o funcionamento automático do *shuttle car* – Shuttle_Load_Unload.

FB Shuttle Manual: Controla o funcionamento, em modo manual, do *shuttle car*.

FB Shuttle Load Unload: Responsável pelo funcionamento, em modo automático, do *shuttle car*. Este *function block* controla a execução das ordens de transporte. Perante uma nova ordem (fornecida pelo bloco Reading_Array), é verificada:

- A presença de uma carga a transportar, a partir do estado do sensor existente no final da linha correspondente à coordenada de origem, presente na ordem recebida (fornecido pelo bloco Order_Sensor);
- A disponibilidade do destino, a partir do sensor colocado no início da linha correspondente à coordenada de destino presente na ordem recebida (fornecido pelo bloco Dest_Sensor e Dest_Cnt).

Caso se verifiquem estas duas condições e o *shuttle car* esteja pronto a ordem é executada. O GRAFCET deste *function block* é apresentado na Figura 39.

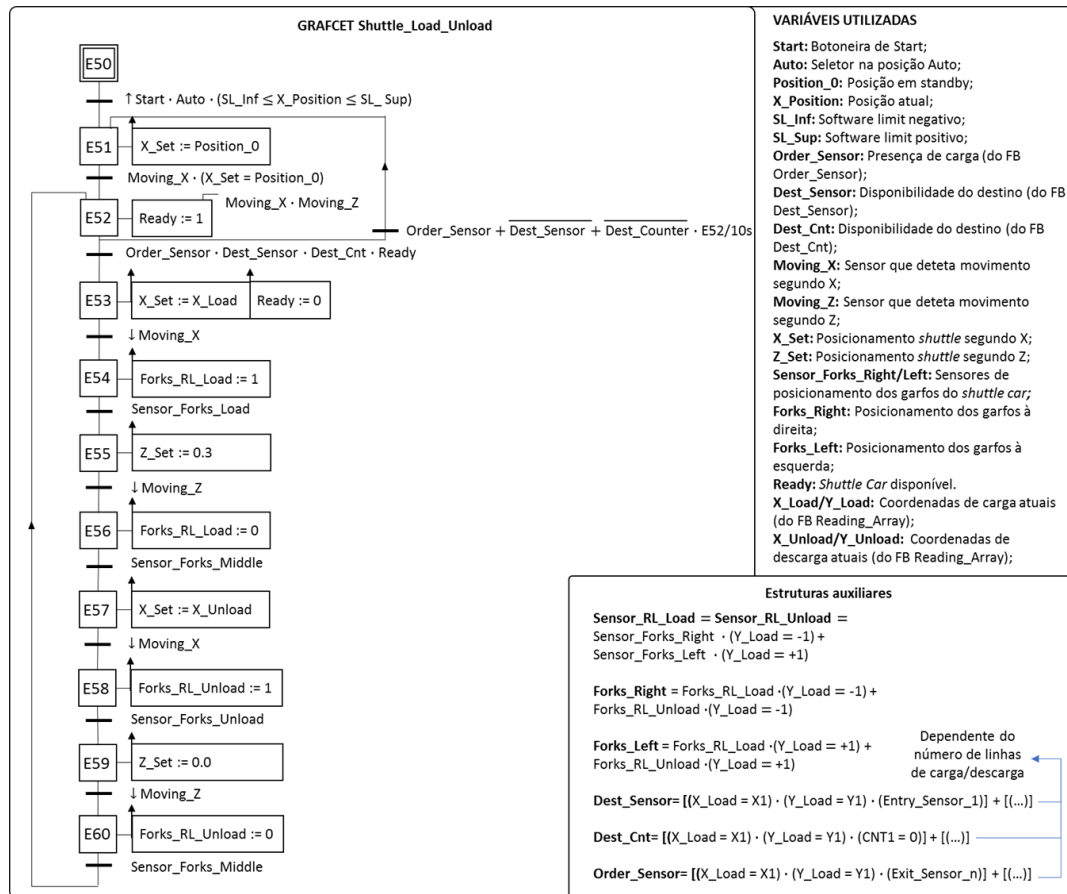


Figura 39 - GRAFCET do *function block* Shuttle_Load_Unload.

FB Dest Sensor: Avalia o estado do sensor colocado no início da linha correspondente às coordenadas de descarga presentes na ordem atual. O estado deste sensor (inativo – com carga; ativo – sem carga) é transmitido ao *function block* que controla o funcionamento automático do *shuttle car*, que avalia se a ordem pode ser executada.

FB Dest Cnt: Avalia o valor do contador do primeiro *conveyor* da linha correspondente às coordenadas de descarga presentes na ordem atual. O valor deste contador (não nulo – com carga; nulo – sem carga) é passado ao *function block* que controla o funcionamento automático do *shuttle car*, que avalia se a ordem pode ser executada.

FB Order Sensor: Avalia o estado do sensor existente no final da linha correspondente às coordenadas de carga da ordem atual. O estado deste sensor (inativo – com carga; ativo – sem carga) é passado ao bloco que controla o funcionamento automático do *shuttle car*, que avalia se a ordem pode ser executada.

Para além dos *function blocks* descritos anteriormente, foi necessário adicionar um outro que controlasse a simulação no FactoryIO®. Este é descrito seguidamente.

FB #Simulation: Controla a colocação de paletes nas linhas de carga do ambiente gráfico desenvolvido em FactoryIO®, ativando os emissores existentes consoante as coordenadas presentes em cada ordem lida (a partir do FB Reading_List). Perante uma nova ordem lida, verifica-se a disponibilidade do *conveyor* correspondente para receber a carga (a partir do

sensor no início da linha de entrada). Caso esteja disponível, a paleta é emitida; caso contrário, aguarda até que esta condição se verifique.

Organization Blocks

De modo a instanciar os *function blocks* necessários ao controlo do sistema foi necessária a criação do *organization block* Main. Para cada instância são criadas *instance data blocks* (DBs) que contêm os valores das variáveis declaradas no FB. Para além disso, o OB foi também utilizado para fazer a passagem direta das variáveis e *inputs/outputs* necessários a cada FB.

Note-se que o número e tipo de *function blocks* a instanciar no OB está dependente das variações na configuração do sistema a controlar.

Global Database Blocks

As *global database blocks* descritas em seguida foram utilizadas de modo a acomodar variáveis necessárias ao controlo do sistema.

Global Data: Contém diversos parâmetros essenciais ao controlo do *shuttle car* e *conveyors*, entre os quais limites de software, posição de standby, fatores de redução de velocidade, tempos de carga/descarga e de transição entre *conveyors*.

Orders (Figura 40): Contém as ordens de transporte a serem executadas pelo *shuttle car*. Cada ordem contém dois pares de coordenadas: coordenadas de origem/carga (X_Origin_Conv e Y_Origin) e coordenadas de destino/descarga (X_Destination_Conv e Y_Destination). A implementação da lista no TIA Portal® é realizada recorrendo um *struct*, um tipo de variável que permite agrupar várias variáveis – neste caso, vetores.

Orders				
	Name	Data type	Offset	Start value
1	Static			
2	Orders	Array[0..99] ...	0.0	
3	Orders[0]	Struct	0.0	
4	X_Origin_Conv	Real	0.0	1.3095
5	X_Destination_...	Real	4.0	9.405
6	Y_Origin	Int	8.0	-1
7	Y_Destination	Int	10.0	-1
8	Orders[1]	Struct	12.0	
9	X_Origin_Conv	Real	12.0	3.333
10	X_Destination_...	Real	16.0	9.405
11	Y_Origin	Int	20.0	-1
12	Y_Destination	Int	22.0	-1
13	Orders[2]	Struct	24.0	
14	Orders[3]	Struct	36.0	
15	Orders[4]	Struct	48.0	

Figura 40 – Lista de ordens de transporte a cumprir pelo *shuttle car*.

Tagtables

Seguidamente, descrevem-se as *tagtables* criadas para acomodar os *inputs* e *outputs* necessários ao controlo do sistema.

Inputs: Contém os inputs físicos (ex: sensores, botoneiras, etc.) e respetivo endereçamento no PLC.

Outputs: Contém os outputs físicos (ex: atuadores dos *conveyors* e *shuttle car*, sinalizadores, etc.) e respetivo endereçamento no PLC.

5 Aplicação desenvolvida

Neste capítulo é feita uma descrição pormenorizada da aplicação desenvolvida para a geração automática do programa de controlo do *shuttle car*.

Em primeiro lugar descreve-se, de uma forma geral, o objetivo pretendido pela aplicação. Em seguida expõe-se detalhadamente a arquitetura da aplicação e respetiva interface gráfica, de modo a que se possam compreender as funções por esta implementadas. Por último, explora-se aprofundadamente o método utilizado para implementar a geração de código - a manipulação de documentos XML, apresentando-se alguns exemplos de geração de código.

5.1 Objetivo da aplicação

Como referido anteriormente, o objetivo da aplicação desenvolvida é gerar o código do programa de controlo do *shuttle car*, de modo a flexibilizar a programação destes sistemas, minimizando o seu tempo de desenvolvimento.

A aplicação desenvolvida permite gerar o programa de controlo do *shuttle car* de acordo com as variantes descritas no ponto 3.5. Deste modo, é possível controlar sistemas semelhantes com *layouts* distintos, como os ilustrados nas Figuras 41 e 42.

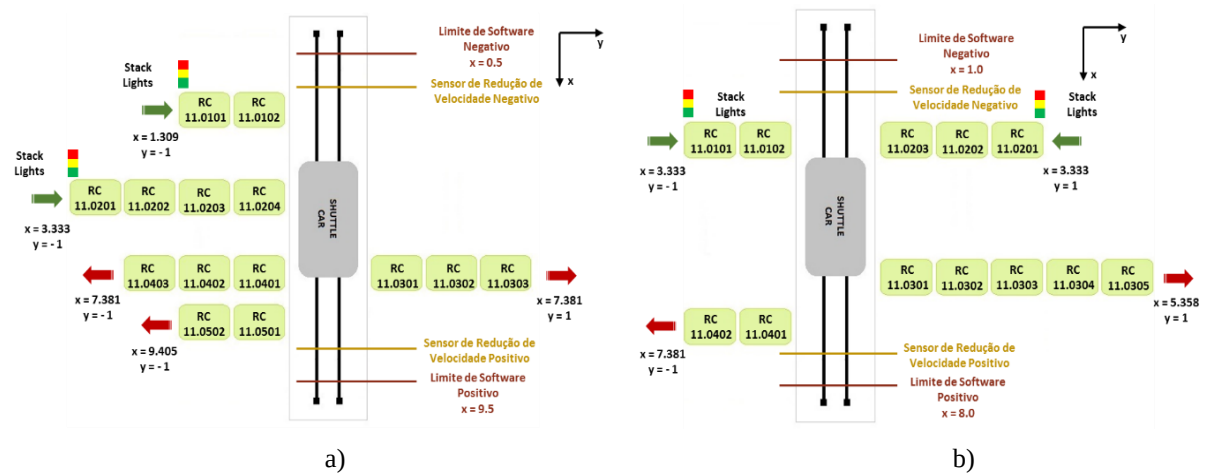


Figura 41 – Sistemas com *layouts* distintos: a) Sistema A; b) Sistema B.



Figura 42 – Sistemas com *layouts* distintos em ambiente de simulação: a) Sistema A; b) Sistema B.

5.2 Arquitetura da aplicação

A Figura 43 representa a arquitetura de alto nível da aplicação desenvolvida para a geração e manipulação de código para o programa de controlo do *shuttle car*.

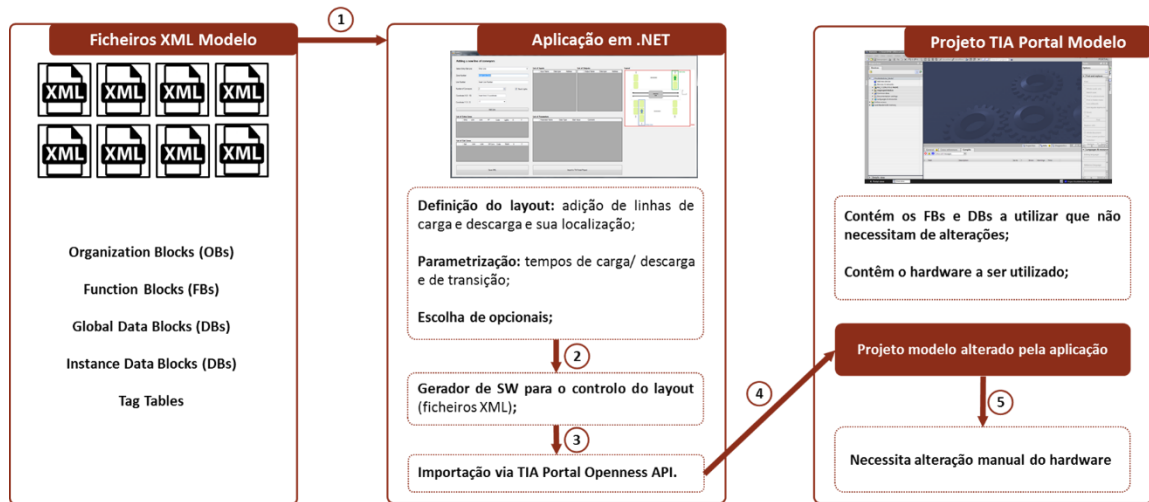


Figura 43 – Arquitetura da aplicação.

Na primeira fase, a aplicação lê um conjunto de documentos XML modelo, correspondentes aos vários elementos de software a manipular: OBs, FBs, *global* DBs e *instance* DBs e *tag*tables.

Em seguida, a partir da interface gráfica definem-se os parâmetros desejados no que diz respeito ao *layout*, tempos de carga/descarga/transição e opcionais. Estes dados, permitem que a aplicação defina a informação a introduzir nos ficheiros XML modelo, de modo a que estes contenham o código necessário ao controlo do sistema a implementar.

Uma vez produzidos os documentos XML necessários, é feita a sua importação para o projeto modelo (Figura 44). Este contém, unicamente, elementos de software que não necessitam de alterações e o hardware a utilizar (PLC e cartas de I/O).

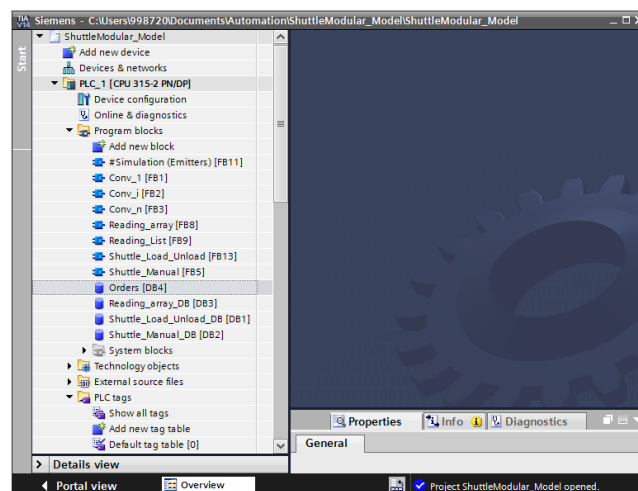


Figura 44 – Projeto modelo no TIA Portal.

Após a importação dos ficheiros XML, obtém-se o programa necessário ao controlo do sistema, num estado quase final, sendo apenas necessário efetuar algumas alterações manuais ao nível das cartas de I/O incorporadas no projeto.

5.3 Interface da aplicação

A Figura 45 ilustra a interface gráfica da aplicação desenvolvida, contendo uma descrição dos seus constituintes.

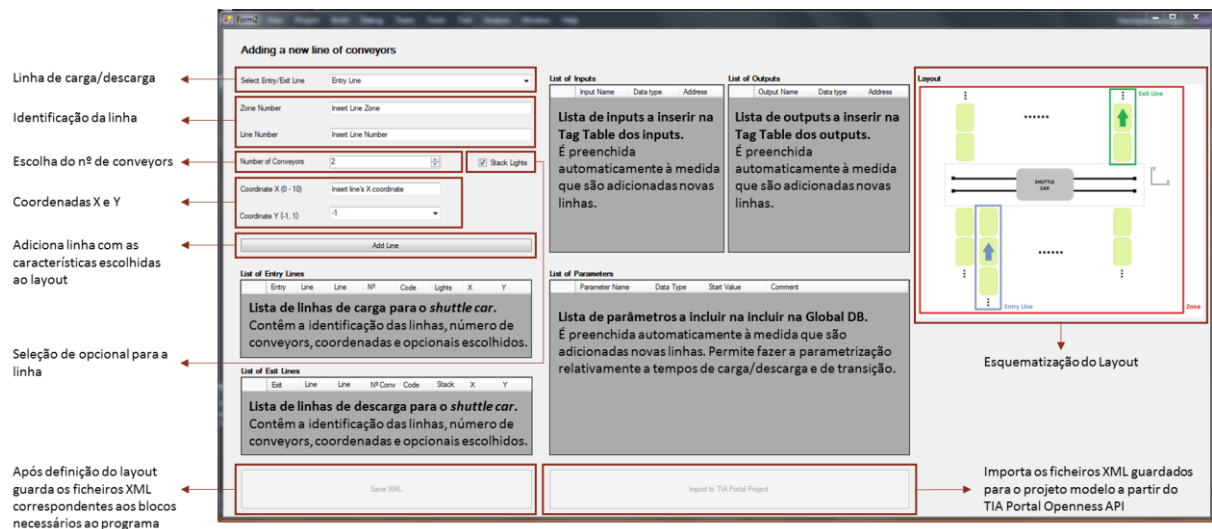


Figura 45 – Interface gráfica da aplicação.

A interface gráfica permite definir a configuração do sistema a controlar a partir da introdução de linhas de carga/descarga no *layout*. Ao adicionar cada nova linha é necessário indicar um conjunto de informações, entre as quais:

- Tipo de linha (carga/descarga);
- Identificação da linha (número da zona e da linha);
- Número de *conveyors* que constituem a linha;
- Pretensão de incluir opcionais (sinalizadores coluna (*stacklights*) – Figura 32b);
- Posicionamento da linha, a partir das coordenadas X e Y.

Além disso, a interface gráfica permite também visualizar as linhas introduzidas e, caso necessário, editá-las ou apagá-las a partir das listas de linhas.

Adicionalmente, de cada vez que é introduzida uma nova linha, é possível acompanhar a inserção de variáveis nas listas de *Inputs*, *Outputs* e parâmetros, sendo possível editá-las.

Por último, a interface gráfica permite fazer a gravação dos documentos XML modificados e a importação dos mesmos para o projeto modelo a partir dos botões *SaveXML* e *Import to TIA Portal Project*, respetivamente.

5.4 Funcionamento da aplicação

Nesta secção, é analisado o comportamento da aplicação no cumprimento das funções de adição de novas linhas, de gravação dos documentos XML e de importação dos mesmos para o projeto modelo do TIA Portal®.

5.4.1 Adição de uma nova linha

A função de adição de uma nova linha está acessível a partir do botão *Add Line*, mediante o preenchimento da informação relativa à linha (Secção 5.3). Na prática, esta função não produz qualquer alteração nos documentos XML modelo, limitando-se unicamente a preencher as listas (de linhas, *inputs*, *outputs* e parâmetros) incluídas na interface gráfica. A informação presente nessas listas contém a informação na qual a aplicação se baseia para efetuar a totalidade das manipulações nos documentos XML modelo.

De modo a compreender estas as manipulações, é necessário explorar as implicações da adição de novas linhas de carga e descarga no programa de controlo. Recomenda-se a consulta da Secção 4.2.2 para uma melhor compreensão do raciocínio descrito seguidamente.

A adição de uma nova linha de descarga ao sistema implica (Figura 46):

1. Alteração do FB Dest_Sensor, dado que passa a existir uma nova coordenada de destino e, conseqüentemente, um novo sensor cujo estado tem de ser verificado;
2. Alteração do FB Dest_Cnt, dado que passa a existir uma nova coordenada de destino e, conseqüentemente, um novo contador cujo valor tem de ser avaliado;
3. Alteração das *instance* DBs correspondentes ao FB Dest_Sensor e FB_Dest_Cnt;
4. Instanciação dos FBs Dest_Sensor e Dest_Cnt e identificação das respectivas DBs no OB;
5. Alteração das *instance* DBs correspondentes aos FBs que controlam os *conveyors* da nova linha;
6. Instanciação dos FBs que controlam os *conveyors* (Conv_1, Conv_i e Conv_n) no OB;
7. Adição de *inputs* e *outputs* às respectivas *tagtables* e de novos parâmetros à *database* Global Data, como consequência da inclusão de novos *conveyors*.

As manipulações dos ficheiros XML correspondentes aos pontos 1, 6 e 7 são apresentadas no capítulo 5.5.4.2.

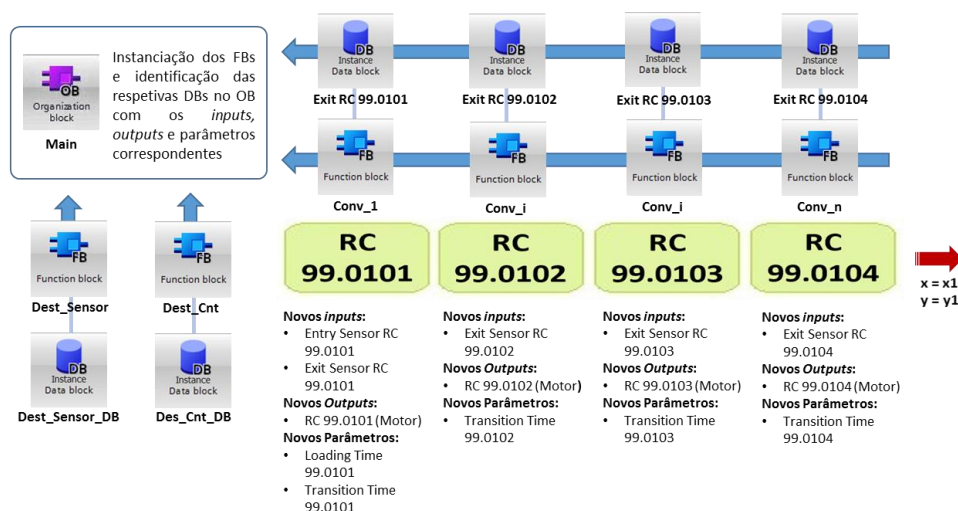


Figura 46 – Esquematização das manipulações necessárias na adição de uma nova linha de descarga.

Por sua vez, a adição de uma linha de carga implica o cumprimento dos seguintes passos:

1. Alteração do FB Order_Sensor, dado que passa a existir uma nova coordenada de origem e, consequentemente, um novo sensor cujo estado tem de ser verificado;
2. Alteração da *instance* DB correspondente ao FB Order_Sensor;
3. Instanciação do FB Order_Sensor e identificação da respetiva DB no OB;
4. Aplicação dos passos 5,6 e 7 descritos para as linhas de descarga.

As manipulações dos ficheiros XML correspondentes aos pontos 1 a 4 são semelhantes às verificadas para as linhas de descarga, não sendo por isso exploradas.

5.4.2 Gravação dos ficheiros XML

A função de gravação, disponível a partir do botão *Save XML*, executa as alterações nos vários documentos XML a partir das informações contidas nas listas de linhas, *inputs*, *outputs* e parâmetros.

Uma vez manipulados todos os ficheiros necessários a aplicação permite, a partir de um diálogo, seleccionar um destino para a gravação dos mesmos. Após a gravação, a função de importação para o projeto modelo fica disponível.

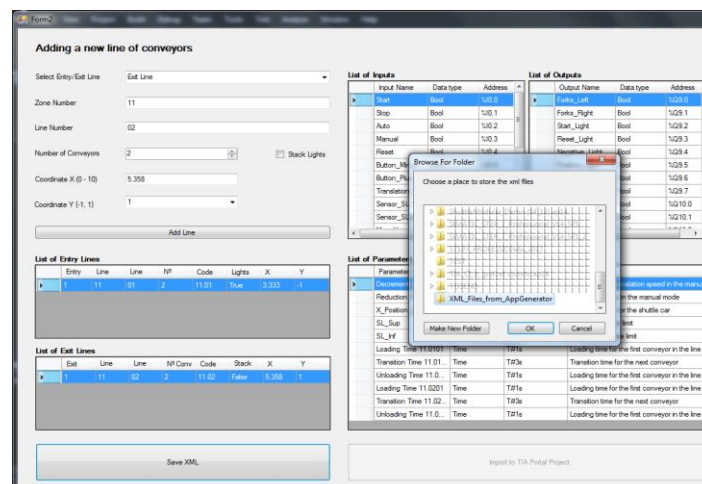
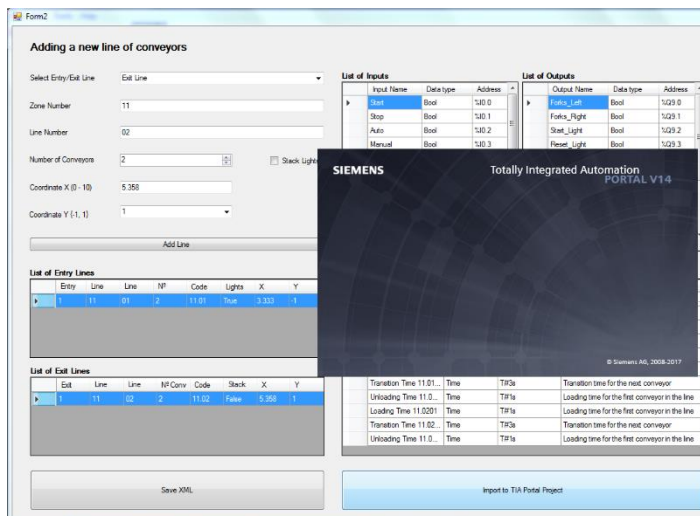


Figura 47 – Gravação dos documentos XML manipulados.

5.4.3 Importação dos ficheiros XML para o projeto modelo

A função de importação, acessível a partir do botão *Import to TIA Portal Project*, executa as seguintes ações: iniciação de uma instância do TIA Portal®, abertura do projeto modelo e importação dos documentos XML manipulados. A sua implementação foi conseguida a partir da interface *Public API* do TIA Portal Openness®, apresentando-se, em seguida, os métodos utilizados para cada ação:

- StartTIA(): Iniciação de uma nova instância do TIA Portal® (Figura 48);



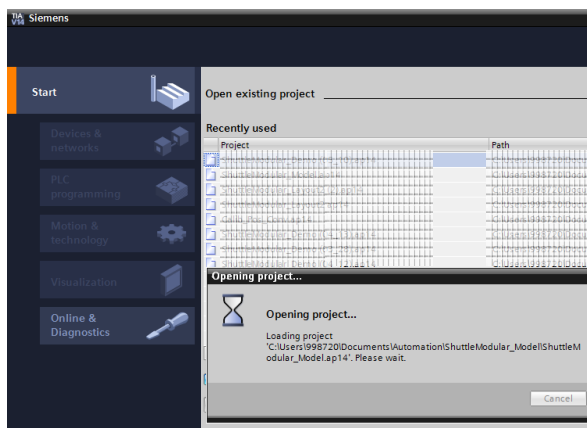
a)

```
private void StartTIA()
{
    MyTiaPortal = new TiaPortal(TiaPortalMode.
        WithUserInterface); // Nova instância
}
```

b)

Figura 48 – Iniciação de uma nova instância do TIA Portal: a) Interface gráfica; b) Método utilizado.

- OpenProject(): Abertura do projeto modelo (Figura 49);



a)

```
private void OpenProject(string ProjectPath)
{
    try
    {
        MyProject = MyTiaPortal.Projects.Open(new
            FileInfo(ProjectPath));
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.
            ToString());
    }
}
```

b)

Figura 49 – Abertura do projeto modelo: a) TIA Portal; b) Método utilizado.

- ImportBlocks(): Importação dos blocos (OBs, FBs, *global* DBs e *instance* DBs).

A implementação deste método exige o acesso à área *Program Blocks* do TIA Portal®, correspondente à área de destino deste tipo de elementos de software. Como se pode constatar a partir da observação da Figura 50, em primeiro lugar acedeu-se ao *Device* (PLC_1), seguido do acesso ao seu software (*plcSoftware*), que contém a área *ProgramBlocks*. O percurso de acesso à área foi deduzido a partir dos modelos da *Public API*, referidos no capítulo 3.4.1.1.

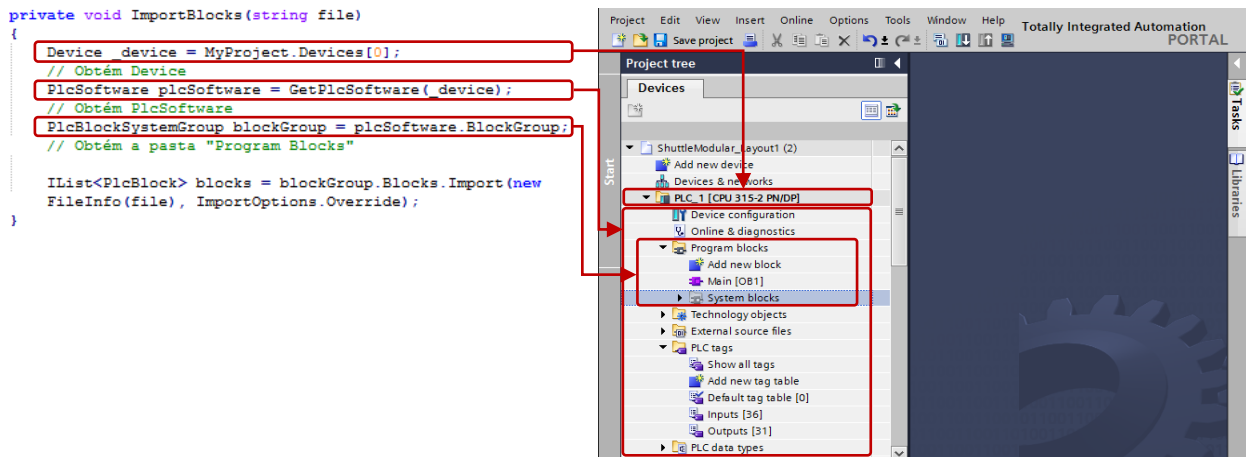


Figura 50 – Acesso e importação para a área *Program Blocks*.

- ImportTagTable(): Importação das *tagtables*.

A implementação deste método exige o acesso ao destino *PLC Tags*, visível na janela do TIA Portal® disposta na Figura 50. O método de acesso (Figura 51) foi semelhante ao utilizado para aceder à área *ProgramBlocks*.

```
private void ImportTagTable(string file)
{
    Device _device = MyProject.Devices[0]; // Obtém Station
    PlcSoftware plcSoftware = GetPlcSoftware(_device); // Obtém PlcSoftware
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup; // Obtém grupo PLCtagTables
    PlcTagTableComposition tagTables = plcTagTableSystemGroup.TagTables; // Obtém a pasta "PLCTags"
    tagTables.Import(new FileInfo(file), ImportOptions.Override);
}
```

Figura 51 – Acesso e importação para a área *PLC Tags*.

5.5 Geração de código a partir da manipulação de documentos XML

Nesta secção aborda-se a forma a partir da qual foi possível gerar o programa de controlo do *shuttle car* para a configuração desejada, a partir da manipulação de documentos XML.

Assim, em primeiro lugar descreve-se a estrutura geral dos documentos XML do TIA Portal®, bem como o processo de criação dos documentos XML modelo utilizados pela aplicação. É também apresentado o método utilizado na manipulação destes documentos, bem como alguns exemplos das manipulações efetuadas pela aplicação.

5.5.1 Estrutura dos documentos XML do TIA Portal®

Uma vez que se optou por desenvolver uma aplicação que gerasse código a partir da manipulação dos documentos XML do TIA Portal®, foi necessário analisar a sua estrutura.

Deste modo, analisa-se em seguida a estrutura geral dos documentos XML do TIA Portal®, bem como a estrutura de um *organization block* em particular. A estrutura dos documentos XML correspondentes aos restantes elementos de software do TIA Portal® utilizados pela aplicação é exposta no Anexo A.

Estrutura Geral

A estrutura dos documentos XML de um elemento de software de um projeto do TIA Portal® depende do tipo de exportação executada, do tipo de elemento exportado (OB, FB, DB, etc.) e da linguagem de programação utilizada.

Os documentos XML do TIA Portal® são compostos por duas áreas principais (Figura 52): *DocumentInfo*, que contém a informação relativa à criação do documento e aos produtos instalados no TIA Portal®; e *SW.CodeBlock*, que descreve o elemento de software exportado.

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <DocumentInfo>
  <SW.CodeBlock ID="0">
</Document>
```

Figura 52 – Estrutura geral dos ficheiros XML do TIA Portal.

As áreas do tipo *SW.CodeBlock* diferem na sua designação e conteúdo conforme o elemento de software que descrevem (Figura 53), sendo constituídas por outras duas áreas: *AttributeList* e *ObjectList*. A primeira, *AttributeList*, contém informação geral sobre o elemento exportado e estabelece a sua interface, caso esta exista (*inputs*, *outputs*, constantes, etc.) (Figura 54). A segunda, *ObjectList*, contém diversos objetos identificados por IDs, como por exemplo: comentários, títulos, *networks*, *tags*, entre outros. A cada objeto usado corresponde um ID com um valor distinto (entre 0 e 2147483647).

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <DocumentInfo>
  <SW.Tags.PlcTagTable ID="0">
</Document>
```

a)

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <DocumentInfo>
  <SW.Blocks.GlobalDB ID="0">
</Document>
```

b)

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <DocumentInfo>
  <SW.Blocks.FB ID="0">
</Document>
```

c)

Figura 53 – Designações da área *SW.CodeBlock* para os componentes: a) *TagTable*; b) *Global DB*; c) *FB*.

Name	Data type	Comment
Input		
START_M	Bool	Start the Motor
STOP_M	Bool	Stop the Motor
MAINTENANCE MODE	Bool	Activates the Maintenance M
ROTATION SPEED	Real	


```
<Interface><Sections xmlns="http://www.siemens.com/automat
<Section Name="Input">
  <Member Name="START_M" Datatype="Bool" Accessibility="Pr
  <Comment>
    <MultiLanguageText Lang="en-US">Start the Motor</Mu
    <MultiLanguageText Lang="de-DE">Startet den Motor</M
  </Comment>
</Member>
  <Member Name="STOP_M" Datatype="Bool" Accessibility="Pr
  <Member Name="MAINTENANCE MODE" Datatype="Bool" Access
  <Member Name="ROTATION SPEED" Datatype="Real" Accessibil
</Section>
<Section Name="Output">
<Section Name="InOut" />
<Section Name="Temp">
<Section Name="Constant" />
<Section Name="Return" />
</Sections></Interface>
```

Figura 54 – Interface de um componente.

Estrutura do documento XML de um OB

Na Figura 55 dispõe-se e analisa-se a estrutura do documento XML do *organization block* do programa desenvolvido.

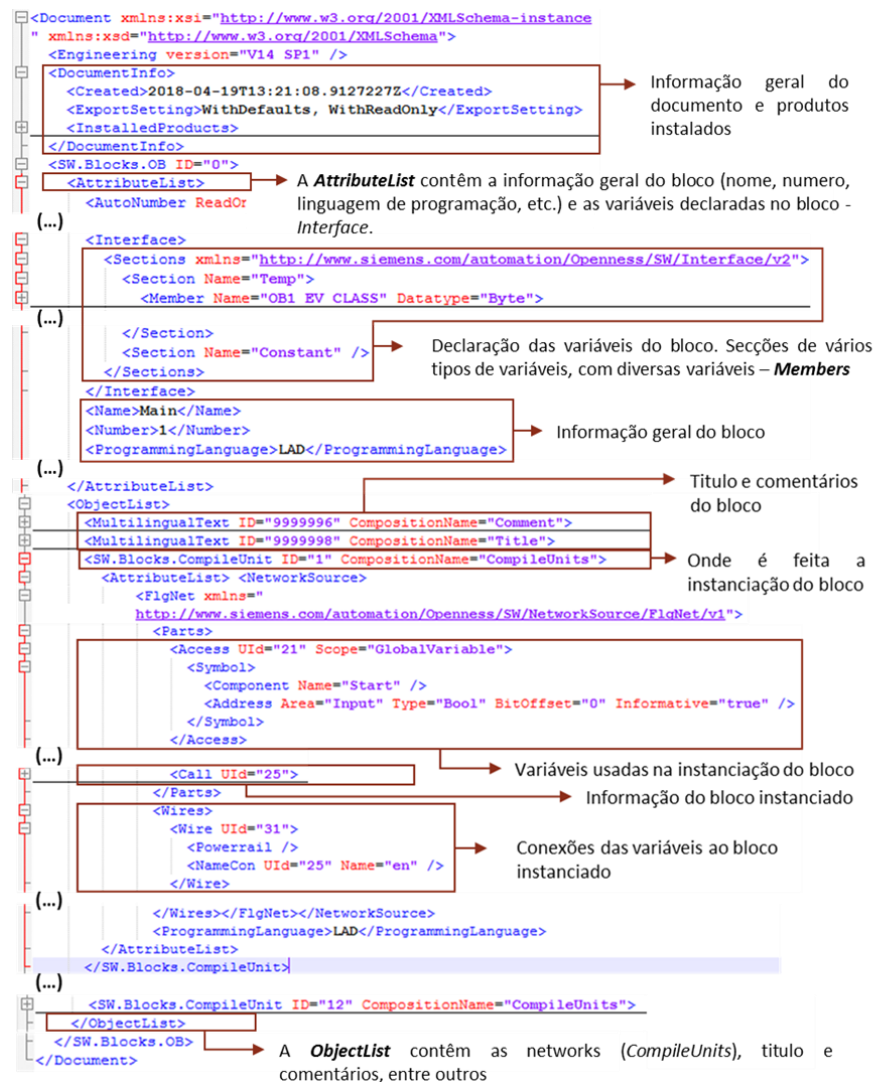


Figura 55 – Estrutura do XML de um *organization block*.

Como se pode observar, neste caso, a área *SW.CodeBlock* é designada por *SW.Blocks.OB*, sendo composta por outras duas áreas: *Attribute List* e *Object List* (Figura 55). A primeira, *Attribute List*, identifica a interface do bloco. A segunda, *ObjectList*, é maioritariamente composta por *networks* (*SW.CompileUnit*), que contêm o código presente no *organization block*.

Cada área *SW.CompileUnit* pode conter uma *ObjectList* e uma *AttributeList*. A *ObjectList*, caso exista, contém os comentários da *network*. A *AttributeList* incorpora: a *NetworkSource*, composta pelas *Parts* (variáveis e comandos usados na *network*) e pelos *Wires* (relação entre as variáveis e os comandos) e a *ProgrammingLanguage*, que identifica a linguagem de programação utilizada na *network*.

A forma como as *Parts* e os *Wires* se relacionam encontra-se esquematizada na Figura 56, para um caso geral. É importante realçar que as *Parts* e *Wires* têm *Uids* associados, que devem ser únicos dentro da *SW.CompileUnit* (valores entre 21 e 2147483647).

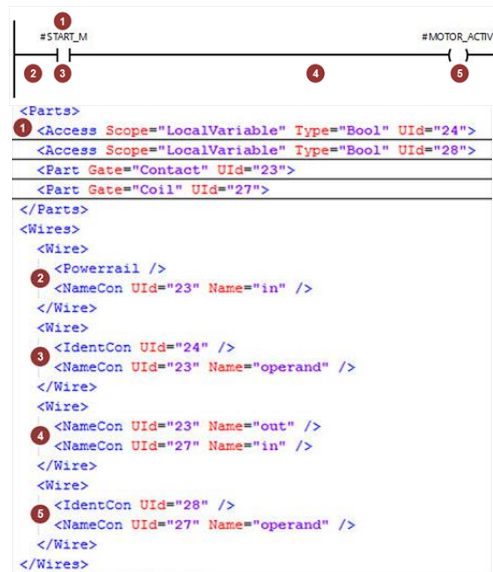


Figura 56 – Exemplo geral da relação entre *Parts* e *Wires*.

5.5.2 Documentos XML modelo

Como descrito na Secção 5.2, a aplicação executa a geração de código a partir da manipulação de documentos XML modelo correspondentes aos diversos elementos de software de um projeto do TIA Portal® (OBs, FBs, *instance* DBs, *global* DBs e *tagtables*).

Os documentos XML modelo foram concebidos em dois passos distintos. Em primeiro lugar, exportaram-se os documentos XML correspondentes aos vários elementos de software do programa de controlo desenvolvido na Secção 4.2, usando a aplicação *DemoOpenness* (Secção 3.4.1.3). Seguidamente, efetuou-se a sua alteração manual, de modo a que contivessem apenas informação geral e independente do sistema a controlar. Na maioria dos casos, os documentos XML modelo correspondem a elementos de software vazios, isto é, sem qualquer código.

A título de exemplo, a Figura 57 ilustra o documento XML modelo para o FB *Order_Sensor*. Através da análise da Figura 57a pode observar-se que, na área *Interface* existe apenas uma variável, a qual se mantém imutável qualquer que seja o sistema a controlar. Na *ObjectList* (Figura 57b) pode verificar-se que existem duas áreas de comentários e uma *network* que se encontra vazia e, por isso, sem código. Os restantes modelos para os diversos elementos de software do TIA Portal® são expostos no Anexo A.



Figura 57 – Documento XML modelo para o FB *Order_Sensor*.

5.5.3 Método de Manipulação dos ficheiros XML do TIA Portal®

Na Secção 3.4.1.4 exploraram-se diversos métodos utilizados na manipulação de um documento XML através de uma aplicação em C#. Seguidamente faz-se uma análise comparativa das várias alternativas disponíveis, tendo em conta os requisitos necessários à sua utilização na aplicação de geração de código.

O uso da classe *XmlSerialize* permite utilizar os XML *Schemas* fornecidos pela Siemens, que descrevem a estrutura dos documentos XML correspondentes aos vários elementos de software de um projeto do TIA Portal®. Deste modo, é possível gerar automaticamente as classes correspondentes aos documentos XML e assim aceder às diversas áreas dos mesmos, permitindo a sua criação e edição, a partir dos métodos da classe *XmlSerialize*.

A utilização das classes *XmlTextReader* e *XmlTextWriter* foi desde logo excluída, uma vez que não permite a edição de documentos XML.

Por sua vez, a classe *XmlDocument*, inicialmente considerada, revelou grandes desvantagens relativamente à classe *XmlSerialize*, nomeadamente a necessidade de desenvolver inúmeros métodos, que, a partir do *XPath*, permitissem aceder às diversas áreas dos documentos XML. Com a classe *XmlSerialize* e as classes geradas, esta funcionalidade fica automaticamente garantida.

De acordo com esta informação, optou-se por utilizar a classe *XmlSerialize* e por gerar automaticamente as classes necessárias à sua utilização a partir do *Developer Command Prompt* (Figura 58). A geração das classes foi executada com os XML *Schemas* *SW.InterfaceSections.v2.xsd* e *SW.PlcBlocks.LADFB.xsd*. Devido à presença da instrução *include* nestes ficheiros, no processo de geração das classes são ainda incluídos outros três XML *Schemas* (*SW.PlcBlocks.CompileUnitCommon.xsd*, *SW.PlcBlocks.Access.xsd* e *SW.Common.xsd*). O resultado deste processo é uma classe (ficheiro .cs) pronto a ser importado para o *Microsoft Visual Studio* e que contém um conjunto de classes que permitem o acesso as diferentes áreas dos documentos XML do TIA Portal®.

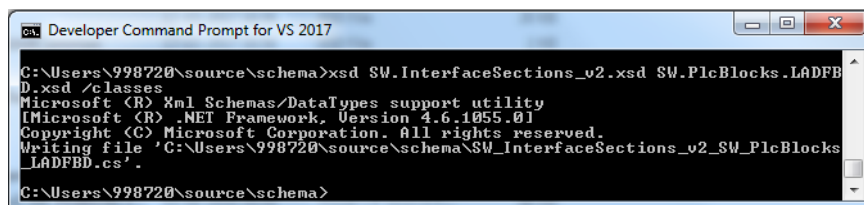


Figura 58 – Criação das classes necessárias à manipulação dos ficheiros XML via *XmlSerialize*.

No entanto, dada a imprecisão deste método, foi necessário criar novas classes e efetuar algumas alterações nas classes geradas, de modo a poder aceder às várias áreas dos documentos XML do TIA Portal®.

Os XML *Schemas* utilizados na geração das classes, bem como as classes utilizadas para os documentos XML dos vários elementos de software do TIA Portal® podem ser encontradas no Anexo A.

5.5.4 Manipulações efetuadas nos Documentos XML do TIA Portal®

Nesta secção apresentam-se os elementos de software do projeto desenvolvido em TIA Portal® cujos documentos XML são manipulados pela aplicação de geração de código desenvolvida. São também expostos diversos exemplos das manipulações efetuadas nos diversos elementos de software, bem como os seus resultados no que diz respeito ao programa de controlo resultante.

5.5.4.1 Componentes Manipulados

Tal como descrito no Capítulo 5.2, a aplicação de geração de código manipula diversos documentos XML modelo. O resultado é a alteração do conteúdo de vários elementos de software do projeto desenvolvido em TIA Portal®, de modo a que este se adeque ao sistema a controlar. Na Figura 59 apresenta-se uma esquematização dos elementos de software manipulados pela aplicação desenvolvida.

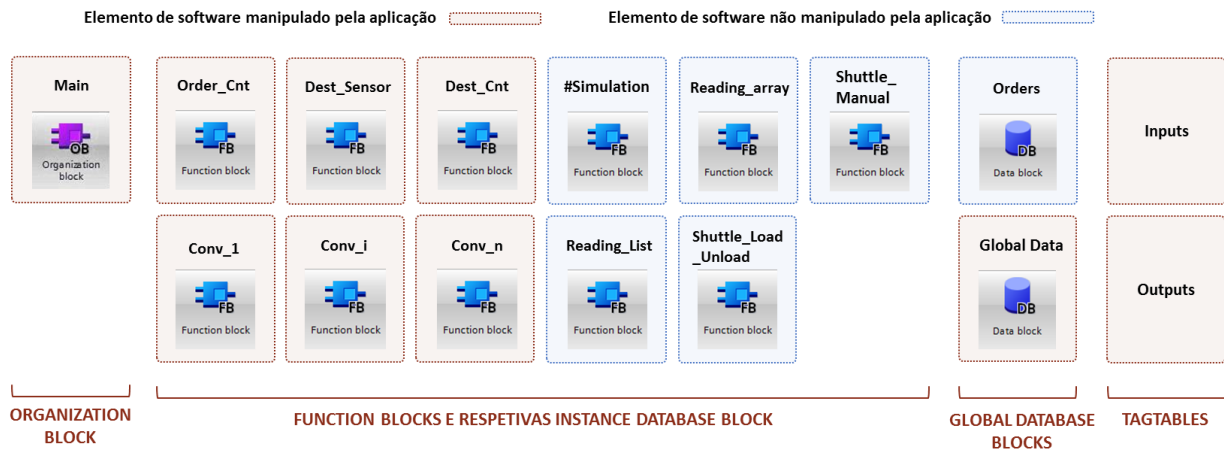


Figura 59 – Elementos de software do projeto desenvolvido manipulados pela aplicação.

5.5.4.2 Exemplos

De acordo com o descrito no ponto 5.4.1, as manipulações a efetuar nos elementos de software do programa desenvolvido e, conseqüentemente, nos documentos XML modelo correspondentes, estão dependentes do número de linhas de carga e descarga adicionadas ao sistema.

Assim, tendo em consideração os *layouts* A e B (Figura 41), caracterizados por diferentes números de linhas, apresentam-se em seguida diversos exemplos das manipulações efetuadas.

Instanciação dos FBs responsáveis pelo controlo dos *conveyors* no OB

O documento XML do OB produzido pela aplicação contém as instanciações dos FBs responsáveis pelo controlo de todos os *conveyors* de um sistema. No entanto, na Figura 60, são apenas expostas as instanciações dos FBs que controlam a linha dois do sistema A (Figura 41), composta por quatro *conveyors*. Na análise deste exemplo recomenda-se a consulta da Figura 46.

Como se pode constatar a partir da análise da Figura 60, a aplicação instancia quatro FBs, um para cada *conveyor* da linha. De modo a implementar cada instância, a aplicação executa diversas manipulações no documento XML, de entre as quais:

1. Criação de uma nova *network* (*SW.Blocks.CompileUnit*);
2. Identificação do FB a instanciar e a correspondente *instance* DB (*Call*);
3. Adição das variáveis usadas na instância (*Access*);
4. Estabelecimento das conexões entre as variáveis e o bloco instanciado (*Wires*).

Concluindo, o documento XML produzido pela aplicação terá tantas instâncias quanto o número de *conveyors* presentes no sistema a controlar. Este facto permite que o *organization block* resultante possa ser utilizado em qualquer *layout* do sistema.



Figura 60 – Instanciação dos FBs responsáveis pelo controlo dos *conveyors* no OB.

Alteração do FB Dest_Sensor

Tal como descrito no ponto 5.4.1, o FB Dest_Sensor necessita de alterações de cada vez que é adicionada uma nova linha de descarga. Deste modo, os documentos XML produzidos pela aplicação para os sistemas A e B (Figura 41) serão distintos, dado que contêm diferentes números de linhas de descarga. Consequentemente, os FB Dest_Sensor resultantes serão também eles diferentes.

Para o sistema A (três linhas de descarga), expõem-se respetivamente, nas Figuras 61 e 62, a interface do bloco e as variáveis e comandos usados na *network*. No caso do sistema B (duas linhas de descarga), os mesmos elementos são ilustrados nas Figuras 63 e 64. Estas ilustrações estabelecem um paralelismo entre o documento XML produzido pela aplicação e o resultado obtido no programa.

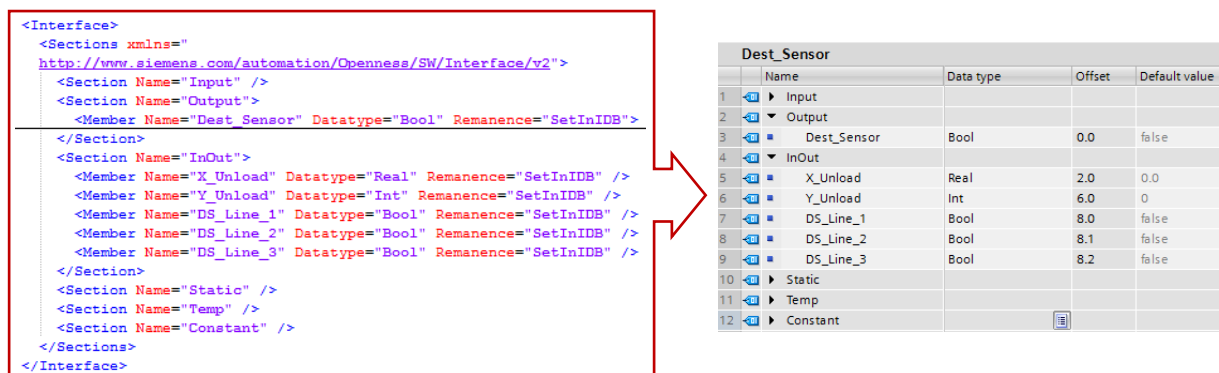


Figura 61 – Interface do FB Dest_Sensor (sistema A): XML produzido (à esquerda); resultado obtido no programa (à direita).

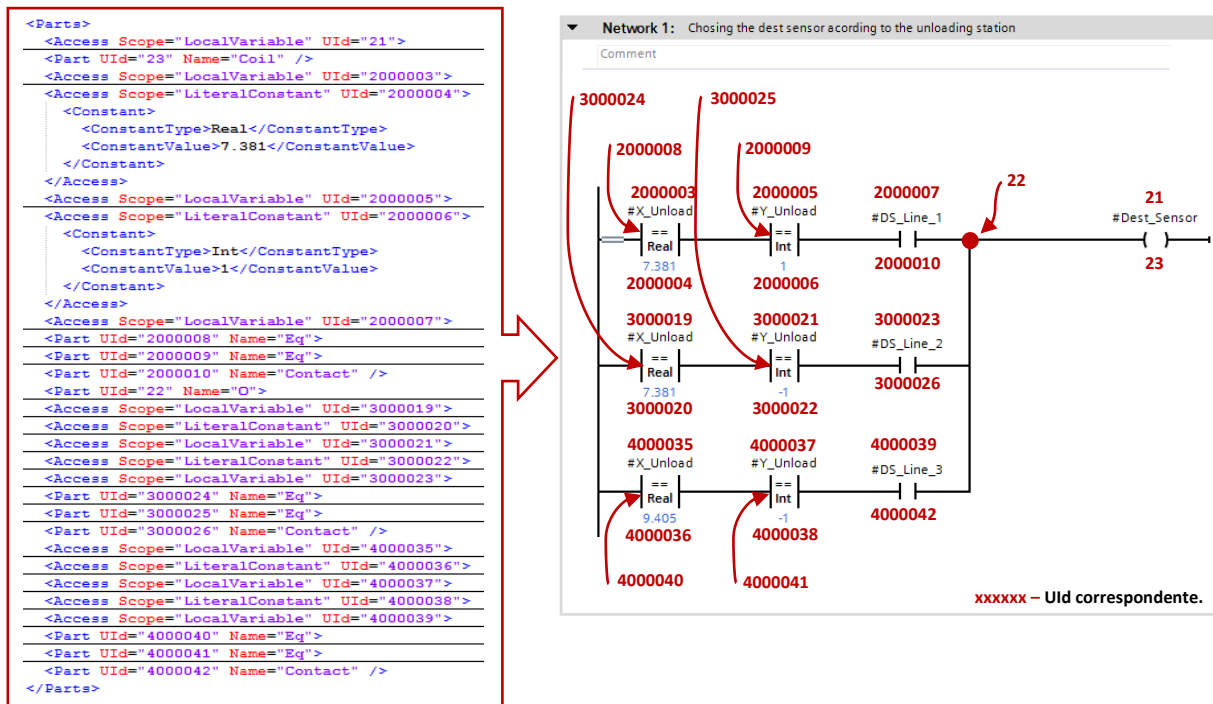


Figura 62 – Variáveis/comandos usados na *network* (sistema A): XML produzido (esquerda); resultado obtido no programa (direita).

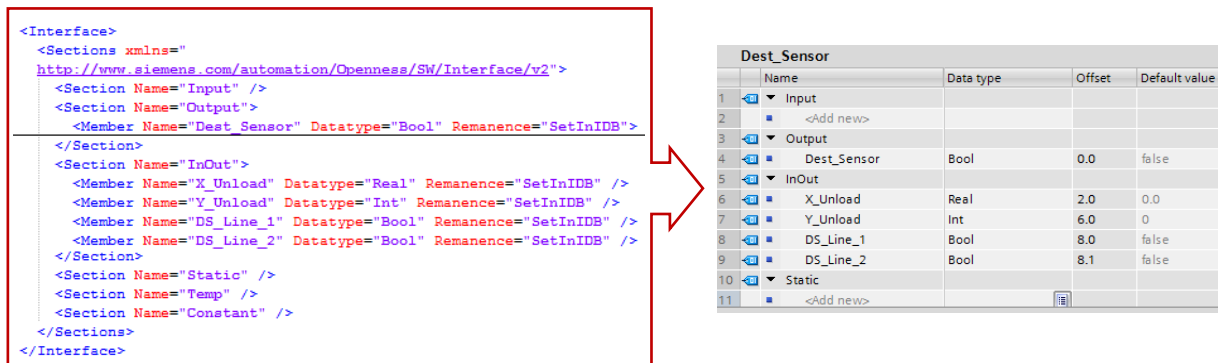


Figura 63 – Interface do FB Dest_Sensor (sistema B): XML produzido (à esquerda); resultado obtido no programa (à direita).

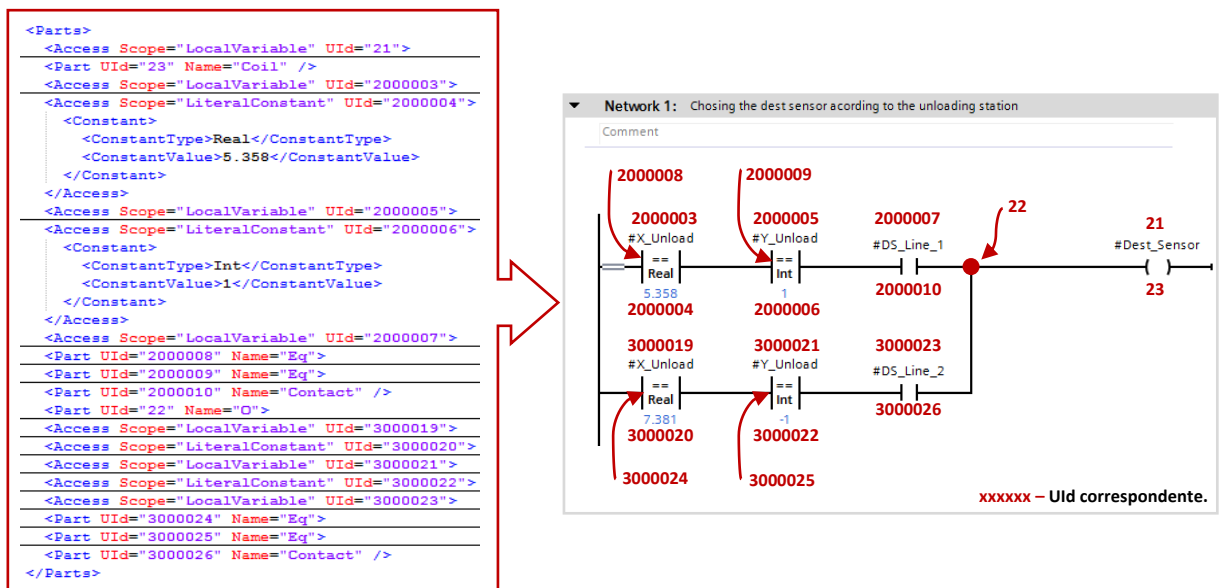


Figura 64 – Variáveis/comandos usados na *network* (sistema B): XML produzido (esquerda); resultado obtido no programa (direita).

Comparando as Figuras 61 e 63, é possível observar as alterações efetuadas pela aplicação na interface do bloco, a partir do documento XML produzido. Consoante o número de linhas de descarga do sistema, são adicionadas em igual número, no documento XML variáveis *DS_Line*. Estas correspondem à introdução de um novo sensor de destino que necessita de ser verificado durante o cumprimento das ordens de transporte por parte do *shuttle car* (Capítulo 5.4.1).

As alterações descritas no parágrafo anterior são complementadas pela adição das variáveis e comandos utilizados na *network* (*Parts*) e pelo estabelecimento das suas relações (*Wires*). Nas Figuras 62 e 64, é possível constatar que, conforme o número de linhas de descarga, são adicionadas variáveis (*Access*) e comandos (*Part*) em diferentes números.

Em suma, a aplicação permite, a partir dos documentos XML produzidos, gerar o código contido no bloco FB Dest_Sensor, podendo este ser aplicado facilmente a vários sistemas com *layouts* distintos.

Adição de *inputs* à *tagtable* Inputs

De acordo com o descrito no capítulo 5.4.1, é necessário adicionar *inputs* à respetiva *tagtable*, de cada vez que é adicionada uma nova linha ao *layout* do sistema. De modo a facilitar a compreensão dos *inputs* adicionados nos exemplos seguintes, recomenda-se a análise da Figura 46 (Capítulo 5.4.1).

Nas Figuras 65 e 66, estabelece-se o paralelismo entre os documentos XML produzidos pela aplicação e as *tagtables* resultantes, para os sistemas A e B (Figura 41). A partir da comparação das mesmas é possível constatar que, perante *layouts* distintos, com diferentes números de linhas e de *conveyors*, a quantidade de variáveis adicionada é também diferente. Deste modo, conclui-se que a aplicação é capaz de produzir *tagtables* consoante o *layout* do sistema, possibilitando a sua aplicação em múltiplas situações.



```

<SW.Tags.PlcTagTable ID="0">
  <AttributeList>
    <Name>Inputs</Name>
  </AttributeList>
  <ObjectList>
    (...)
    <SW.Tags.PlcTag ID="16" CompositionName="Tags">
      <SW.Tags.PlcTag ID="17" CompositionName="Tags">
        <AttributeList>
          <DataTypeName>Bool</DataTypeName>
          <ExternalAccessible>true</ExternalAccessible>
          <ExternalVisible>true</ExternalVisible>
          <ExternalWritable>true</ExternalWritable>
          <LogicalAddress>%I2.0</LogicalAddress>
          <Name>EntrySensor 11.0101</Name>
        </AttributeList>
      </SW.Tags.PlcTag>
    </SW.Tags.PlcTag>
    <SW.Tags.PlcTag ID="18" CompositionName="Tags">
    <SW.Tags.PlcTag ID="19" CompositionName="Tags">
    <SW.Tags.PlcTag ID="20" CompositionName="Tags">
    <SW.Tags.PlcTag ID="21" CompositionName="Tags">
    <SW.Tags.PlcTag ID="22" CompositionName="Tags">
    <SW.Tags.PlcTag ID="23" CompositionName="Tags">
    <SW.Tags.PlcTag ID="24" CompositionName="Tags">
    <SW.Tags.PlcTag ID="25" CompositionName="Tags">
    <SW.Tags.PlcTag ID="26" CompositionName="Tags">
    <SW.Tags.PlcTag ID="27" CompositionName="Tags">
    <SW.Tags.PlcTag ID="28" CompositionName="Tags">
    <SW.Tags.PlcTag ID="29" CompositionName="Tags">
    <SW.Tags.PlcTag ID="30" CompositionName="Tags">
    <SW.Tags.PlcTag ID="31" CompositionName="Tags">
    <SW.Tags.PlcTag ID="32" CompositionName="Tags">
    <SW.Tags.PlcTag ID="33" CompositionName="Tags">
    <SW.Tags.PlcTag ID="34" CompositionName="Tags">
    <SW.Tags.PlcTag ID="35" CompositionName="Tags">
  </ObjectList>
</SW.Tags.PlcTagTable>

```

Inputs			
	Name	Data type	Address
	(111 Variáveis fixas 111)		
16	ByPass	Bool	%I1.7
17	EntrySensor 11.0101	Bool	%I2.0
18	ExitSensor 11.0101	Bool	%I2.1
19	ExitSensor 11.0102	Bool	%I2.2
20	EntrySensor 11.0201	Bool	%I2.3
21	ExitSensor 11.0201	Bool	%I2.4
22	ExitSensor 11.0202	Bool	%I2.5
23	ExitSensor 11.0203	Bool	%I2.6
24	ExitSensor 11.0204	Bool	%I2.7
25	EntrySensor 11.0301	Bool	%I3.0
26	ExitSensor 11.0301	Bool	%I3.1
27	ExitSensor 11.0302	Bool	%I3.2
28	ExitSensor 11.0303	Bool	%I3.3
29	EntrySensor 11.0401	Bool	%I3.4
30	ExitSensor 11.0401	Bool	%I3.5
31	ExitSensor 11.0402	Bool	%I3.6
32	ExitSensor 11.0403	Bool	%I3.7
33	EntrySensor 11.0501	Bool	%I4.0
34	ExitSensor 11.0501	Bool	%I4.1
35	ExitSensor 11.0502	Bool	%I4.2

Figura 65 – *Tagtable* Inputs (sistema A): XML produzido (esquerda); *tagtable* resultante (direita).

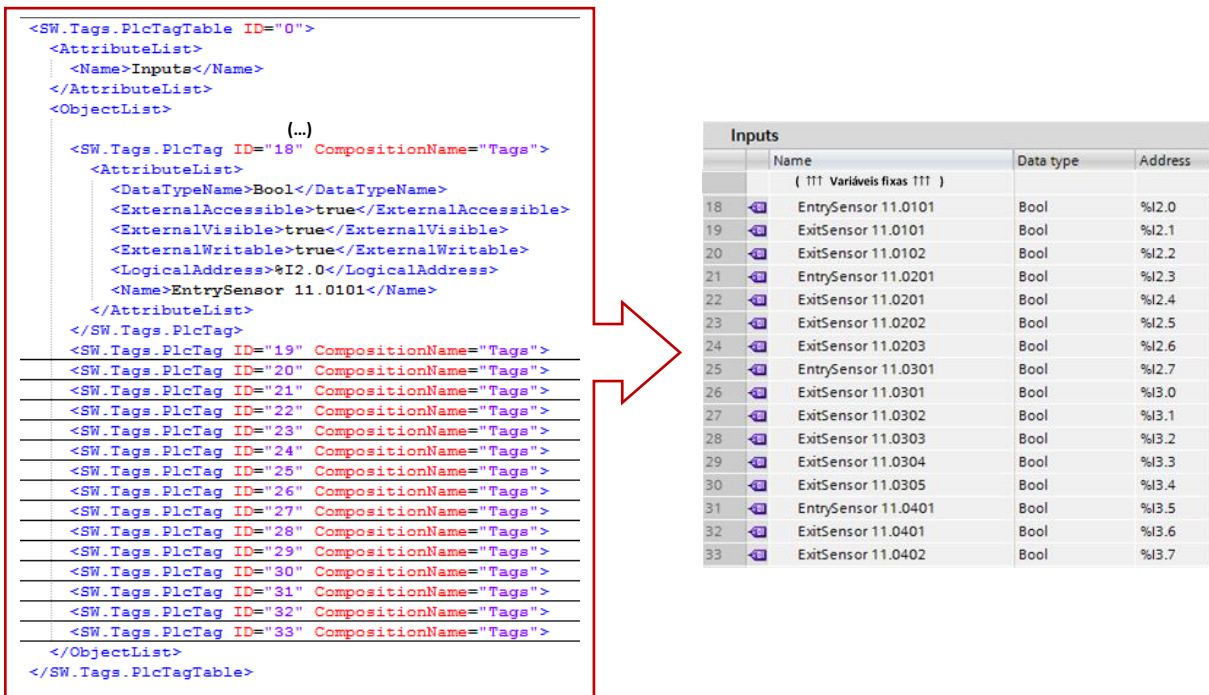


Figura 66 – Tagtable Inputs (sistema B): XML produzido (esquerda); tagtable resultante (direita).

6 Conclusões e perspectivas de trabalho futuro

6.1 Conclusões

O tema da presente dissertação apelava ao desenvolvimento de uma aplicação de software para a geração automática de código para o PLC de controlo de um *shuttle car*.

O carácter inovador da temática da dissertação, e as restrições relativas à compatibilidade da ferramenta de geração de código a desenvolver com as práticas aplicadas na Consoveyo, S.A., criaram incerteza quanto à concretização da mesma. Nesse sentido, optou-se por generalizar o tema da dissertação e explorar as possibilidades existentes relativas à flexibilização da programação de PLCs.

De acordo com a bibliografia consultada, a flexibilização da programação de PLCs podia ser atingida recorrendo a várias abordagens, de entre as quais: a geração de código e a parametrização de variáveis através da comunicação com o PLC. Para além disso, identificaram-se múltiplas ferramentas que permitiam abordar cada uma das alternativas.

Avaliando as potencialidades das ferramentas identificadas e, de modo a explorar cada uma das abordagens, seleccionaram-se o TIA Portal Openness® (geração de código) e o Sharp 7 (parametrização de variáveis através da comunicação com o PLC).

A partir da exploração aprofundada das funcionalidades e capacidades das ferramentas seleccionadas, concluiu-se que a geração de código era a abordagem mais vantajosa. O TIA Portal Openness® permitia implementar uma vasta gama de funcionalidades, nomeadamente o acesso a múltiplas áreas do TIA Portal® (dados de projeto, PLCs e HMIs) e a possibilidade de exportação e importação de informação acerca do hardware e software, a partir de documentos XML. Um simples teste manual, provou que seria possível gerar o software contido num projeto do TIA Portal® (OBs, FBs, DBs, etc.) a partir de uma aplicação externa, que manipulasse documentos XML perante uma dada configuração do sistema a controlar.

Uma vez que o programa de controlo dos *shuttle cars* da Consoveyo, S.A. fora concebido no SIMATIC STEP 7 e, dado que o TIA Portal Openness® opera apenas com o TIA Portal®, tornou-se necessário criar um novo programa nesta plataforma. Este foi desenvolvido tentando replicar o funcionamento dos sistemas da Consoveyo, S.A. e abrangendo o maior número possível de variantes. Para além disso, foi concebido de uma forma modular e eficiente, de modo a que pudesse ser aplicado à geração de código.

Com base na modularidade do código desenvolvido e nas funcionalidades do TIA Portal Openness®, concebeu-se uma aplicação que gera o programa necessário ao controlo do *shuttle car*, perante a seleção de um número relevante de variantes. Deste modo, concluiu-se que era possível controlar sistemas com diferentes configurações a partir do programa produzido pela aplicação.

A seleção das variantes e da configuração desejadas, a partir da interface gráfica da aplicação, permitiu gerar o software contido no programa modelo, quer ao nível de variáveis utilizadas e seus valores, identificações e endereçamento, quer ao nível do código incluído. Geraram-se *organization blocks*, *function blocks*, *database blocks* e *tagtables* a partir da manipulação dos documentos XML correspondentes a cada um destes elementos de software.

6.2 Trabalhos futuros

Esta dissertação constitui uma primeira abordagem à geração automática de código para PLCs, havendo naturalmente múltiplas possibilidades para a continuidade deste projeto. Deste modo, são de seguida sugeridos trabalhos futuros, que visam melhorar o funcionamento global desta aplicação, assim como a expansão do seu horizonte de aplicabilidade.

Sugere-se a expansão das funcionalidades da aplicação, de modo a que seja também possível gerar o hardware necessário para o controlo da configuração desejada. Assim, seria possível, por exemplo, escolher o PLC a incorporar no projeto, bem como incluir automaticamente as cartas de I/O necessárias.

Além disso, a gama de variantes abrangida pela aplicação poderá também ser dilatada, com vista a possibilitar a geração de programas para uma maior diversidade de configurações do *shuttle car*.

Adicionalmente, a aplicação poderá também ser escalada a outros equipamentos, no sentido de agilizar e flexibilizar a programação de um armazém automático.

Por último, seria vantajoso desenvolver uma aplicação semelhante com base no programa desenvolvido pela Consoveyo, S.A., de modo a que pudessem ser gerados os programas para o sistema real concebido pela empresa, e para que o seu comportamento pudesse ser testado. Tal facto torna obrigatória a existência de um programa de controlo em TIA Portal®, e que este tenha a modularidade necessária à implementação de uma ferramenta como esta.

Referências

- Alphonsus, Ephrem Ryan, e Mohammad Omar Abdullah. 2016. “A Review on the Applications of Programmable Logic Controllers (PLCs).” *Renewable and Sustainable Energy Reviews* 60 (Julho). Elsevier: 1185–1205. <https://doi.org/10.1016/j.rser.2016.01.025>.
- Andersson, Mikael, e Erik Helander. 2010. “Automatic Generation of PLC Programs Using Automation Designer.”
- Baron, Luc. 1999. “AN OPTIMAL SURFACING POST-PROCESSOR MODULE FOR 5-AXES CNC MILLING MACHINES.” *Third International Conference on Industrial Automation*.
- Consoveyo. 2017a. “Consoveyo.” 2017. <http://www.consoveyo.com/en/homepage.html>.
- Consoveyo. 2017b. *Harmonia Em Movimento*.
- Consoveyo. 2017c. “Manual De Operação Do Sistema de Periféricos.”
- Critical Software. 2018. “Verificação E Validação de Sistemas Críticos.” 2018. <https://www.criticalsoftware.com/pt/what-we-do/safety-critical-verification-and-validation>.
- Darvas, Daniel, Enrique Blanco Vinuela, e Istvan Majzik. 2016. “PLC Code Generation Based on a Formal Specification Language.” *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, 389–96. <https://doi.org/10.1109/INDIN.2016.7819191>.
- Frey, G., e L. Litz. 2000. “Formal Methods in PLC Programming.” *2000 IEEE International Conference on Systems, Man and Cybernetics* 4: 2431–36. <https://doi.org/10.1109/ICSMC.2000.884356>.
- Güttel, Knut, Peter Weber, e Alexander Fay. 2008. “Automatic Generation of PLC Code beyond the Nominal Sequence.” *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 1277–84. <https://doi.org/10.1109/ETFA.2008.4638565>.
- Hackworth, John R, e Frederick D Hackworth. 2004. *Programmable Logic Controllers : Programming Methods and Applications*.
- Han, Kwan Hee. 2010. “Object-Oriented Modeling, Simulation and Automatic Generation of PLC Ladder Logic.”
- Hasdemir, İ. Tolga, e Salman Kurtulan. 2006. *Automatic Plc Code Generation Using Matlab. IFAC Proceedings Volumes*. Vol. 39. IFAC. <https://doi.org/10.3182/20060926-3-PL-4904.00022>.
- IDC Technologies. 2002. *Industrial Programming Using IEC 61131-3 for PLCs*.
- International Electrotechnical Commission. 2018. “IEC 61131:2018 SER Series.” 2018. <https://webstore.iec.ch/publication/62427>.
- Mathworks. 2017. “Simulink PLC Coder.” 2017. <https://www.mathworks.com/products/sl-plc-coder.html>.
- Mesta Automation. 2015. “Communication with Siemens S7 Plc with C# and S7.Net Plc Driver.” 2015. <https://www.mesta-automation.com/siemens-s7-plc-c-s7-net-plc-driver/>.
- Mesta Automation. 2016. “Snap7: An Open Source Siemens S7 Communication Library.” 2016. <https://www.mesta-automation.com/snap7-open-source-siemens-s7-communication-library/>.
- Parsons, Donald F., Kyle Gress, James M.Dempsey, Joseph Ross, William Parsons, e Stephen Parsons; 1995. AUTOMATED STORAGE AND RETREVAL SYSTEM. 5,379,299, issued

1995.

Siemens. 2016. *TIA Portal Openness: Introduction and Demo Application*. <https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness%3A-introduction-and-demo-application?dti=0&lc=en-WW>.

Siemens. 2017a. *Automating Projects with Scripts*.

Siemens. 2017b. “SIMATIC STEP 7 Professional 2017.” 2017. <https://w3.siemens.com/mcms/simatic-controller-software/en/step7/step7-professional/Pages/Default.aspx>.

Siemens 2017c. “TIA Portal Openness : Generating a Modular Machine with S7-1500,” 1–40.

Siemens 2017d. “Totally Integrated Automation Portal.” 2017. <https://www.siemens.com/global/en/home/products/automation/industry-software/automation-software/tia-portal.html>.

Siemens. 2018. “TIA Portal Openness: Introduction and Demo Application.” 2018. <https://support.industry.siemens.com/cs/document/108716692/tia-portal-openness%3A-introduction-and-demo-application?dti=0&lc=en-WW>.

Snap7. 2016. “Sharp7.” 2016. <http://snap7.sourceforge.net/>.

Steinegger, Michael, Martin Melik-Merkumians, Johannes Zajc, e Georg Schitter. 2016. “Automatic Generation of Diagnostic Handling Code for Decentralized PLC-Based Control Architectures.” *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2016*–Novem. <https://doi.org/10.1109/ETFA.2016.7733694>.

Thapa, Devinder, Chang Mok Park, Sang C. Park, e Gi Nam Wang. 2009. “Auto-Generation of IEC Standard PLC Code Using T-MPSG.” *International Journal of Control, Automation and Systems* 7 (2): 165–74. <https://doi.org/10.1007/s12555-009-0202-z>.

Thramboulidis, Kleanthis. 2012. “IEC 61131 as Enabler of OO and MDD in Industrial Automation.”

Thramboulidis, Kleanthis, e Georg Frey. 2011. “Towards a Model-Driven IEC 61131-Based Development Process in Industrial Automation *” 2011 (Abril): 217–26. <https://doi.org/10.4236/jsea.2011.44024>.

Universidade Técnica de Eindhoven. 2014. “CIF 3 PLC Code Generator.” 2014. <http://cif.se.wtb.tue.nl/tools/cif2plc/index.html#standard-compliance>.

W3C. 2008. “Extensible Markup Language - W3C Recommendation.” 2008. <https://www.w3.org/TR/REC-xml/#sec-origin-goals>.

W3C. 2016. “Extensible Markup Language.” 2016. <https://www.w3.org/XML/>.

W3schools. 2018a. “Introduction to XML.” 2018. https://www.w3schools.com/xml/xml_what.asp.

W3schools. 2018b. “XML Syntax.” 2018. https://www.w3schools.com/xml/xml_syntax.asp.

W3schools. 2018c. “XML Tree.” 2018. https://www.w3schools.com/xml/xml_tree.asp.

Xu, X. W., e S. T. Newman. 2006. “Making CNC Machine Tools More Open, Interoperable and Intelligent - A Review of the Technologies.” *Computers in Industry* 57 (2): 141–52. <https://doi.org/10.1016/j.compind.2005.06.002>.

ANEXO A: Aplicação desenvolvida

Estrutura dos documentos XML dos elementos de software do TIA Portal

Function Block

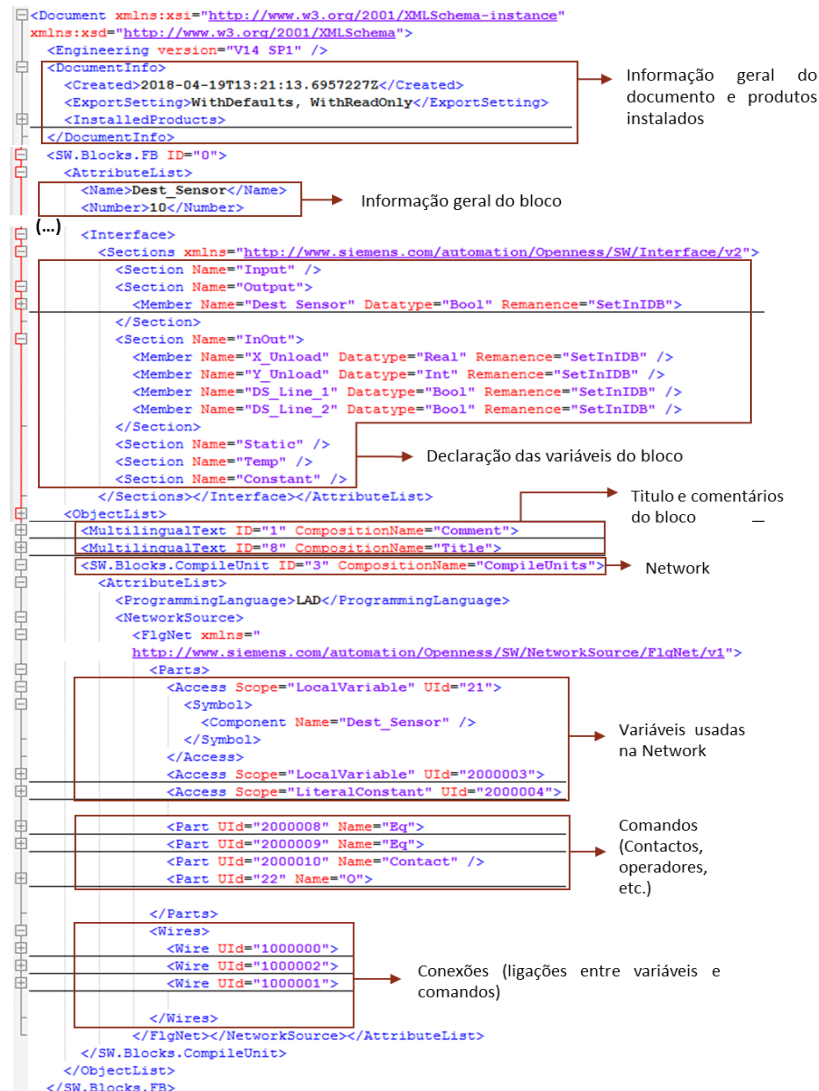


Figura 1 – Estrutura do documento XML de um *function block*.

TagTable

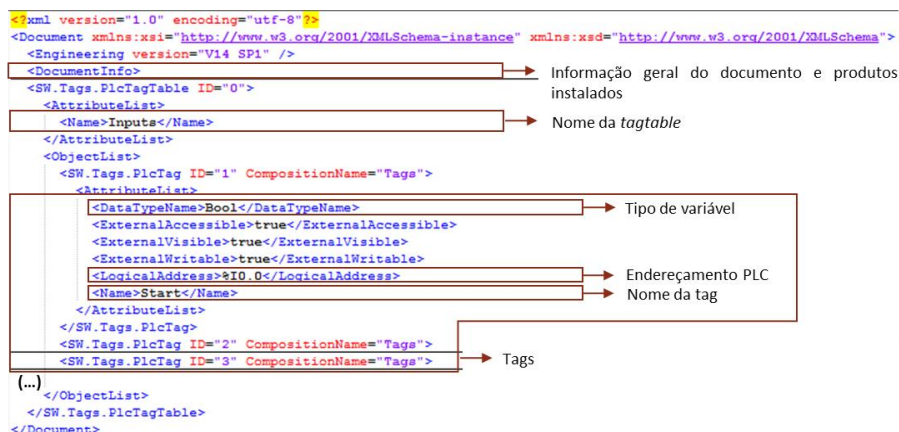


Figura 2– Estrutura do documento XML de uma *tagtable*.

Global Database Block



Figura 3 – Estrutura do documento XML de uma *global database block*.

Instance Database Block

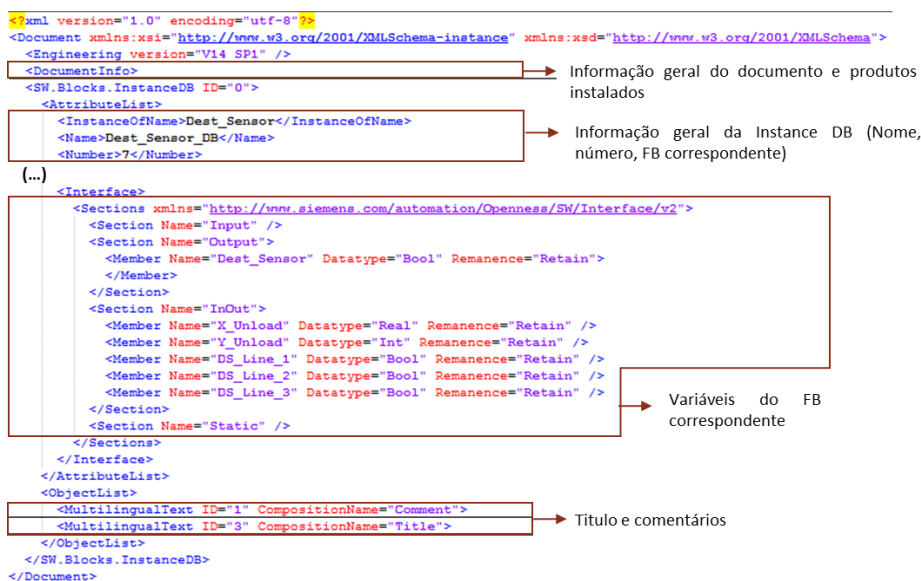


Figura 4 – Estrutura do documento XML de uma *instance database block*.

Documentos XML Modelo dos elementos de software manipulados pela aplicação

Dest_Sensor (FB)

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-04-19T13:21:13.6957227Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
      <Product>
        <DisplayName>Totally Integrated Automation Portal</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </Product>
      <OptionPackage>
        <DisplayName>TIA Portal Openness</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </OptionPackage>
      <Product>
        <DisplayName>STEP 7 Professional</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </Product>
      <Product>
        <DisplayName>WinCC Basic</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </Product>
    </InstalledProducts>
  </DocumentInfo>
  <SW.Blocks.FB ID="0">
    <AttributeList>
      <AutoNumber>true</AutoNumber>
      <CodeModifiedDate ReadOnly="true">2018-04-13T12:33:38.5268935Z</CodeModifiedDate>
      <CompileDate ReadOnly="true">2018-04-13T12:33:38.5268934Z</CompileDate>
      <CreationDate ReadOnly="true">2018-04-05T12:48:08.1976365Z</CreationDate>
      <HeaderAuthor />
      <HeaderFamily />
      <HeaderName />
      <HeaderVersion>0.1</HeaderVersion>
      <Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
        <Section Name="Input" />
        <Section Name="Output">
          <Member Name="Dest_Sensor" Datatype="Bool" Remanence="SetInIDB" />
        </Section>
        <Section Name="InOut">
        </Section>
        <Section Name="Static" />
        <Section Name="Temp" />
        <Section Name="Constant" />
      </Sections></Interface>
      <InterfaceModifiedDate ReadOnly="true">2018-04-11T15:41:42.8373471Z</InterfaceModifiedDate>
      <IsConsistent ReadOnly="true">true</IsConsistent>
      <IsIECCheckEnabled>false</IsIECCheckEnabled>
      <IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
      <ISMultiInstanceCapable ReadOnly="true">true</ISMultiInstanceCapable>
      <IsWriteProtected ReadOnly="true">false</IsWriteProtected>
      <LibraryConformanceStatus ReadOnly="true">The object is library-conformant.</LibraryConformanceStatus>
      <MemoryLayout ReadOnly="true">Standard</MemoryLayout>
      <ModifiedDate ReadOnly="true">2018-04-13T12:33:38.5268935Z</ModifiedDate>
      <Name>Dest_Sensor</Name>
      <Number>10</Number>
      <ParameterModified ReadOnly="true">2018-04-11T15:41:42.8373471Z</ParameterModified>
      <ProgrammingLanguage>LAD</ProgrammingLanguage>
      <StructureModified ReadOnly="true">2018-04-11T15:41:42.8373471Z</StructureModified>
      <UDABlockProperties />
      <UDAEnableTagReadback>false</UDAEnableTagReadback>
    </AttributeList>
    <ObjectList>
      <MultilingualText ID="1" CompositionName="Comment">
        <ObjectList>
          <MultilingualTextItem ID="2" CompositionName="Items">
            <AttributeList>
              <Culture>en-US</Culture>
            </AttributeList>
            <Text />
          </MultilingualTextItem>
        </ObjectList>
      </MultilingualText>
      <SW.Blocks.CompileUnit ID="3" CompositionName="CompileUnits">
        <AttributeList>
          <NetworkSource><FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1">
            <FlgNet></NetworkSource>
          </FlgNet>
          <ProgrammingLanguage>LAD</ProgrammingLanguage>
        </AttributeList>
        <ObjectList>
          <MultilingualText ID="4" CompositionName="Comment">
            <ObjectList>
              <MultilingualTextItem ID="5" CompositionName="Items">

```

```

        <AttributeList>
        <Culture>en-US</Culture>
        <Text />
        </AttributeList>
    </MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="6" CompositionName="Title">
<ObjectList>
    <MultilingualTextItem ID="7" CompositionName="Items">
    <AttributeList>
        <Culture>en-US</Culture>
        <Text>Chosing the dest sensor acording to the unloading station</Text>
    </AttributeList>
    </MultilingualTextItem>
    </ObjectList>
    </MultilingualText>
</ObjectList>
</SW.Blocks.CompileUnit>
<MultilingualText ID="8" CompositionName="Title">
<ObjectList>
    <MultilingualTextItem ID="9" CompositionName="Items">
    <AttributeList>
        <Culture>en-US</Culture>
        <Text />
    </AttributeList>
    </MultilingualTextItem>
    </ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.FB>
</Document>

```

Dest_Sensor (Instance DB)

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-04-19T13:21:14.2707227Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
    </InstalledProducts>
  </DocumentInfo>
  <SW.Blocks.InstanceDB ID="0">
    <AttributeList>
      <AutoNumber>true</AutoNumber>
      <CodeModifiedDate ReadOnly="true">2018-04-13T12:33:39.1119269Z</CodeModifiedDate>
      <CompileDate ReadOnly="true">2018-04-13T12:33:39.1279278Z</CompileDate>
      <CreationDate ReadOnly="true">2018-04-11T15:41:01.4810775Z</CreationDate>
      <HeaderAuthor />
      <HeaderFamily />
      <HeaderName />
      <HeaderVersion>0.1</HeaderVersion>
      <InstanceOfName>Dest_Sensor</InstanceOfName>
      <InstanceOfNumber ReadOnly="true">10</InstanceOfNumber>
      <InstanceOfType ReadOnly="true">FB</InstanceOfType>
      <Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
<Section Name="Input" />
<Section Name="Output">
  <Member Name="Dest_Sensor" Datatype="Bool" Remanence="Retain" />
</Section>
<Section Name="InOut">
  <Member Name="X_Unload" Datatype="Real" Remanence="Retain"></Member>
  <Member Name="Y_Unload" Datatype="Int" Remanence="Retain"></Member>
</Section>
<Section Name="Static" />
</Sections></Interface>
      <InterfaceModifiedDate ReadOnly="true">2018-04-11T15:41:42.8373471Z</InterfaceModifiedDate>
      <IsConsistent ReadOnly="true">true</IsConsistent>
      <IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
      <MemoryLayout ReadOnly="true">Standard</MemoryLayout>
      <ModifiedDate ReadOnly="true">2018-04-13T12:33:39.1119269Z</ModifiedDate>
      <Name>Dest_Sensor_DB</Name>
      <Number>11</Number>
      <ParameterModifiedDate ReadOnly="true">2018-04-11T15:41:42.8373471Z</ParameterModifiedDate>
      <ProgrammingLanguage>DB</ProgrammingLanguage>
      <StructureModifiedDate ReadOnly="true">2018-04-11T15:41:42.8373471Z</StructureModifiedDate>
      <UDABlockProperties ReadOnly="true" />
      <UDAEnableTagReadback ReadOnly="true">false</UDAEnableTagReadback>
    </AttributeList>
    <ObjectList>
      <MultilingualText ID="1" CompositionName="Comment">
    </ObjectList>

```

```

    <MultilingualTextItem ID="2" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text />
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
  <ObjectList>
    <MultilingualTextItem ID="4" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text />
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.InstanceDB>
</Document>

```

Dest_Cnt (FB)

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-04-19T13:21:14.372722Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
    </InstalledProducts>
  </DocumentInfo>
  <SW.Blocks.FB ID="0">
    <AttributeList>
      <AutoNumber>true</AutoNumber>
      <CodeModifiedDate ReadOnly="true">2018-04-13T12:33:36.7707931Z</CodeModifiedDate>
      <CompileDate ReadOnly="true">2018-04-13T12:33:36.770793Z</CompileDate>
      <CreationDate ReadOnly="true">2018-04-13T09:45:09.7643495Z</CreationDate>
      <HeaderAuthor />
      <HeaderFamily />
      <HeaderName />
      <HeaderVersion>0.1</HeaderVersion>
      <Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
        <Section Name="Input" />
        <Section Name="Output">
          <Member Name="Dest_Cnt" Datatype="Bool" Remanence="SetInIDB"><AttributeList><IntegerAttribute Name="Offset"
            Informative="true" SystemDefined="true">0</IntegerAttribute><BooleanAttribute Name="ExternalAccessible"
              SystemDefined="true">true</BooleanAttribute><BooleanAttribute Name="ExternalVisible"
                SystemDefined="true">true</BooleanAttribute><BooleanAttribute Name="ExternalWritable"
                  SystemDefined="true">true</BooleanAttribute><BooleanAttribute Name="UserVisible" Informative="true"
                    SystemDefined="true">true</BooleanAttribute><BooleanAttribute Name="UserReadOnly" Informative="true"
                      SystemDefined="true">false</BooleanAttribute><BooleanAttribute Name="UserDeletable" Informative="true"
                        SystemDefined="true">true</BooleanAttribute></AttributeList></Member>
        </Section>
        <Section Name="InOut">
        </Section>
        <Section Name="Static" />
        <Section Name="Temp" />
        <Section Name="Constant" />
      </Sections></Interface>
      <InterfaceModifiedDate ReadOnly="true">2018-04-13T09:47:40.7024418Z</InterfaceModifiedDate>
      <IsConsistent ReadOnly="true">true</IsConsistent>
      <IsIECCheckEnabled>false</IsIECCheckEnabled>
      <IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
      <ISMultiInstanceCapable ReadOnly="true">true</ISMultiInstanceCapable>
      <IsWriteProtected ReadOnly="true">false</IsWriteProtected>
      <LibraryConformanceStatus ReadOnly="true">The object is library-conformant.</LibraryConformanceStatus>
      <MemoryLayout ReadOnly="true">Standard</MemoryLayout>
      <ModifiedDate ReadOnly="true">2018-04-13T12:33:36.7707931Z</ModifiedDate>
      <Name>Dest_Cnt</Name>
      <Number>12</Number>
      <ParameterModified ReadOnly="true">2018-04-13T09:47:40.7024418Z</ParameterModified>
      <ProgrammingLanguage>LAD</ProgrammingLanguage>
      <StructureModified ReadOnly="true">2018-04-13T09:47:40.7024418Z</StructureModified>
      <UDABlockProperties />
      <UDAEnableTagReadback>false</UDAEnableTagReadback>
    </AttributeList>
    <ObjectList>
      <MultilingualText ID="1" CompositionName="Comment">
        <ObjectList>
          <MultilingualTextItem ID="2" CompositionName="Items">
            <AttributeList>
              <Culture>en-US</Culture>
              <Text />
            </AttributeList>
          </MultilingualTextItem>
        </ObjectList>
      </MultilingualText>
    </ObjectList>
  </SW.Blocks.FB>
</Document>

```

```

        </AttributeList>
    </MultilingualTextItem>
</ObjectList>
</MultilingualText>
<SW.Blocks.CompileUnit ID="3" CompositionName="CompileUnits">
    <AttributeList>
        <NetworkSource><FlgNet xmlns="http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1">
</FlgNet></NetworkSource>
        <ProgrammingLanguage>LAD</ProgrammingLanguage>
    </AttributeList>
    <ObjectList>
        <MultilingualText ID="4" CompositionName="Comment">
            <ObjectList>
                <MultilingualTextItem ID="5" CompositionName="Items">
                    <AttributeList>
                        <Culture>en-US</Culture>
                        <Text />
                    </AttributeList>
                </MultilingualTextItem>
            </ObjectList>
        </MultilingualText>
        <MultilingualText ID="6" CompositionName="Title">
            <ObjectList>
                <MultilingualTextItem ID="7" CompositionName="Items">
                    <AttributeList>
                        <Culture>en-US</Culture>
                        <Text>Chosing the dest counter acording to the unloading station</Text>
                    </AttributeList>
                </MultilingualTextItem>
            </ObjectList>
        </MultilingualText>
    </ObjectList>
</SW.Blocks.CompileUnit>
<MultilingualText ID="8" CompositionName="Title">
    <ObjectList>
        <MultilingualTextItem ID="9" CompositionName="Items">
            <AttributeList>
                <Culture>en-US</Culture>
                <Text />
            </AttributeList>
        </MultilingualTextItem>
    </ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.FB>
</Document>

```

Dest_Cnt (Instance DB)

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
    <Engineering version="V14 SP1" />
    <DocumentInfo>
        <Created>2018-04-19T13:21:14.4017227Z</Created>
        <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
        <InstalledProducts>
            ...
        </InstalledProducts>
    </DocumentInfo>
    <SW.Blocks.InstanceDB ID="0">
        <AttributeList>
            <AutoNumber>true</AutoNumber>
            <CodeModifiedDate ReadOnly="true">2018-04-13T12:33:39.4759477Z</CodeModifiedDate>
            <CompileDate ReadOnly="true">2018-04-13T12:33:39.4999491Z</CompileDate>
            <CreationDate ReadOnly="true">2018-04-13T09:50:54.986827Z</CreationDate>
            <HeaderAuthor />
            <HeaderFamily />
            <HeaderName />
            <HeaderVersion>0.1</HeaderVersion>
            <InstanceOfName>Dest_Cnt</InstanceOfName>
            <InstanceOfNumber ReadOnly="true">12</InstanceOfNumber>
            <InstanceOfType ReadOnly="true">FB</InstanceOfType>
            <Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
                <Section Name="Input" />
                <Section Name="Output">
                    <Member Name="Dest_Cnt" Datatype="Bool" Remanence="Retain" />
                </Section>
                <Section Name="InOut">
                    <Member Name="X_Unload" Datatype="Real" Remanence="Retain" />
                    <Member Name="Y_Unload" Datatype="Int" Remanence="Retain" />
                </Section>
                <Section Name="Static" />
            </Sections></Interface>
            <InterfaceModifiedDate ReadOnly="true">2018-04-13T09:47:40.7024418Z</InterfaceModifiedDate>
            <IsConsistent ReadOnly="true">true</IsConsistent>
            <IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>

```

```
<MemoryLayout ReadOnly="true">Standard</MemoryLayout>
<ModifiedDate ReadOnly="true">2018-04-13T12:33:39.4759477Z</ModifiedDate>
<Name>Dest_Cnt_DB</Name>
<Number>17</Number>
<ParameterModified ReadOnly="true">2018-04-13T09:47:40.7024418Z</ParameterModified>
<ProgrammingLanguage>DB</ProgrammingLanguage>
<StructureModified ReadOnly="true">2018-04-13T09:47:40.7024418Z</StructureModified>
<UDABlockProperties ReadOnly="true" />
<UDAEnableTagReadback ReadOnly="true">false</UDAEnableTagReadback>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
<MultilingualTextItem ID="2" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
<ObjectList>
<MultilingualTextItem ID="4" CompositionName="Items">
<AttributeList>
<Culture>en-US</Culture>
<Text />
</AttributeList>
</MultilingualTextItem>
</ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.InstanceDB>
</Document>
```

Order_Sensor (Instance DB)

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
<Engineering version="V14 SP1" />
<DocumentInfo>
<Created>2018-04-19T13:21:14.2497227Z</Created>
<ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
<InstalledProducts>
...
</InstalledProducts>
</DocumentInfo>
<SW.Blocks.InstanceDB ID="0">
<AttributeList>
<AutoNumber>true</AutoNumber>
<CodeModifiedDate ReadOnly="true">2018-04-13T12:33:39.2769363Z</CodeModifiedDate>
<CompileDate ReadOnly="true">2018-04-13T12:33:39.2879369Z</CompileDate>
<CreationDate ReadOnly="true">2018-04-11T15:27:57.7988441Z</CreationDate>
<HeaderAuthor />
<HeaderFamily />
<HeaderName />
<HeaderVersion>0.1</HeaderVersion>
<InstanceOfName>Order_Sensor</InstanceOfName>
<InstanceOfNumber ReadOnly="true">7</InstanceOfNumber>
<InstanceOfType ReadOnly="true">FB</InstanceOfType>
<Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
<Section Name="Input" />
<Section Name="Output">
<Member Name="Order_Sensor" Datatype="Bool" Remanence="Retain" />
</Section>
<Section Name="InOut">
<Member Name="X_Load" Datatype="Real" Remanence="Retain" />
<Member Name="Y_Load" Datatype="Int" Remanence="Retain" />
</Section>
<Section Name="Static" />
</Sections></Interface>
<InterfaceModifiedDate ReadOnly="true">2018-04-11T15:39:42.7583361Z</InterfaceModifiedDate>
<IsConsistent ReadOnly="true">true</IsConsistent>
<IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
<MemoryLayout ReadOnly="true">Standard</MemoryLayout>
<ModifiedDate ReadOnly="true">2018-04-13T12:33:39.2769363Z</ModifiedDate>
<Name>Order_Sensor_DB</Name>
<Number>10</Number>
<ParameterModified ReadOnly="true">2018-04-11T15:39:42.7583361Z</ParameterModified>
<ProgrammingLanguage>DB</ProgrammingLanguage>
<StructureModified ReadOnly="true">2018-04-11T15:39:42.7583361Z</StructureModified>
<UDABlockProperties ReadOnly="true" />
<UDAEnableTagReadback ReadOnly="true">false</UDAEnableTagReadback>
</AttributeList>
<ObjectList>
<MultilingualText ID="1" CompositionName="Comment">
<ObjectList>
```

```

    <MultilingualTextItem ID="2" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text />
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>
<MultilingualText ID="3" CompositionName="Title">
  <ObjectList>
    <MultilingualTextItem ID="4" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text />
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>
</ObjectList>
</SW.Blocks.InstanceDB>
</Document>

```

Inputs (Tagtable)

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-05-11T14:37:13.2596044Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
      <Product>
        <DisplayName>Totally Integrated Automation Portal</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </Product>
      <OptionPackage>
        <DisplayName>TIA Portal Openness</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </OptionPackage>
      <Product>
        <DisplayName>STEP 7 Professional</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </Product>
      <Product>
        <DisplayName>WinCC Basic</DisplayName>
        <DisplayVersion>V14 SP1</DisplayVersion>
      </Product>
    </InstalledProducts>
  </DocumentInfo>
  <SW.Tags.PlcTagTable ID="0">
    <AttributeList>
      <Name>Inputs</Name>
    </AttributeList>
    <ObjectList>
    </ObjectList>
  </SW.Tags.PlcTagTable>
</Document>

```

Outputs (Tagtable)

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-05-11T14:37:15.6447409Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
      ...
    </InstalledProducts>
  </DocumentInfo>
  <SW.Tags.PlcTagTable ID="0">
    <AttributeList>
      <Name>Outputs</Name>
    </AttributeList>
    <ObjectList>
    </ObjectList>
  </SW.Tags.PlcTagTable>
</Document>

```

Global Data (Global DB)

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-04-19T13:21:11.1407227Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
      ...
    </InstalledProducts>
  </DocumentInfo>
  <SW.Blocks.GlobalDB ID="0">
    <AttributeList>
      <AutoNumber>true</AutoNumber>
      <CodeModifiedDate ReadOnly="true">2018-04-13T12:33:39.3169386Z</CodeModifiedDate>
      <CompileDate ReadOnly="true">2018-04-13T12:33:39.3299394Z</CompileDate>
      <CreationDate ReadOnly="true">2018-03-26T12:48:59.5294752Z</CreationDate>
      <HeaderAuthor />
      <HeaderFamily />
      <HeaderName />
      <HeaderVersion>0.1</HeaderVersion>
      <Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
        <Section Name="Static">
          </Section>
        </Sections></Interface>
      <InterfaceModifiedDate ReadOnly="true">2018-04-12T15:37:18.6885415Z</InterfaceModifiedDate>
      <IsConsistent ReadOnly="true">true</IsConsistent>
      <IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
      <IsOnlyStoredInLoadMemory>false</IsOnlyStoredInLoadMemory>
      <IsPLCDB ReadOnly="true">false</IsPLCDB>
      <IsWriteProtectedInAS>false</IsWriteProtectedInAS>
      <MemoryLayout ReadOnly="true">Standard</MemoryLayout>
      <ModifiedDate ReadOnly="true">2018-04-13T12:33:39.3169386Z</ModifiedDate>
      <Name>Global Data</Name>
      <Number>5</Number>
      <ParameterModified ReadOnly="true">2018-04-12T15:37:18.6885415Z</ParameterModified>
      <ProgrammingLanguage>DB</ProgrammingLanguage>
      <StructureModified ReadOnly="true">2018-04-12T15:37:18.6885415Z</StructureModified>
    </AttributeList>
    <ObjectList>
      <MultilingualText ID="1" CompositionName="Comment">
        <ObjectList>
          <MultilingualTextItem ID="2" CompositionName="Items">
            <AttributeList>
              <Culture>en-US</Culture>
              <Text />
            </AttributeList>
          </MultilingualTextItem>
        </ObjectList>
      </MultilingualText>
      <MultilingualText ID="3" CompositionName="Title">
        <ObjectList>
          <MultilingualTextItem ID="4" CompositionName="Items">
            <AttributeList>
              <Culture>en-US</Culture>
              <Text />
            </AttributeList>
          </MultilingualTextItem>
        </ObjectList>
      </MultilingualText>
    </ObjectList>
  </SW.Blocks.GlobalDB>
</Document>
```

Conv_1 (Instance DB)

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-04-19T13:21:13.7857227Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
      ...
    </InstalledProducts>
  </DocumentInfo>
  <SW.Blocks.InstanceDB ID="0">
    <AttributeList>
```

```

<AutoNumber>true</AutoNumber>
<CodeModifiedDate ReadOnly="true">2018-04-13T12:33:39.0349225Z</CodeModifiedDate>
<CompileDate ReadOnly="true">2018-04-13T12:33:39.0429229Z</CompileDate>
<CreationDate ReadOnly="true">2018-04-11T09:45:59.9027699Z</CreationDate>
<HeaderAuthor />
<HeaderFamily />
<HeaderName />
<HeaderVersion>0.1</HeaderVersion>
<InstanceOfName>Conv_1</InstanceOfName>
<InstanceOfNumber ReadOnly="true">1</InstanceOfNumber>
<InstanceOfType ReadOnly="true">FB</InstanceOfType>
<Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
<Section Name="Input">
<Member Name="Entry_Sensor_1" Datatype="Bool" Remanence="Retain" />
<Member Name="Exit_Sensor_1" Datatype="Bool" Remanence="Retain" />
<Member Name="Start" Datatype="Bool" Remanence="Retain" />
<Member Name="Reset" Datatype="Bool" Remanence="Retain" />
<Member Name="Auto" Datatype="Bool" Remanence="Retain" />
<Member Name="Stop" Datatype="Bool" Remanence="Retain" />
</Section>
<Section Name="Output">
<Member Name="Conveyor_1" Datatype="Bool" Remanence="Retain" />
<Member Name="Red_light" Datatype="Bool" Remanence="Retain" />
<Member Name="Green_light" Datatype="Bool" Remanence="Retain" />
<Member Name="Yellow_light" Datatype="Bool" Remanence="Retain" />
</Section>
<Section Name="InOut">
<Member Name="Transition_Time" Datatype="Time" Remanence="Retain" />
<Member Name="Loading_Time" Datatype="Time" Remanence="Retain" />
<Member Name="CNT_Next_Conv" Datatype="Int" Remanence="Retain" />
</Section>
<Section Name="Static">
<Member Name="E0" Datatype="Bool" Remanence="Retain" />
<Member Name="E1" Datatype="Bool" Remanence="Retain" />
<Member Name="E6" Datatype="Bool" Remanence="Retain" />
<Member Name="E2" Datatype="Bool" Remanence="Retain" />
<Member Name="E3" Datatype="Bool" Remanence="Retain" />
<Member Name="E4" Datatype="Bool" Remanence="Retain" />
<Member Name="E5" Datatype="Bool" Remanence="Retain" />
<Member Name="IEC_Counter_0_Instance" Datatype="CTUD" Version="1.0" Remanence="Retain">
<Sections>
<Section Name="Input">
<Member Name="CU" Datatype="Bool" />
<Member Name="CD" Datatype="Bool" />
<Member Name="R" Datatype="Bool" />
<Member Name="LOAD" Datatype="Bool" />
<Member Name="PV" Datatype="Int" />
</Section>
<Section Name="Output">
<Member Name="QU" Datatype="Bool" />
<Member Name="QD" Datatype="Bool" />
<Member Name="CV" Datatype="Int" />
</Section>
<Section Name="InOut" />
<Section Name="Static">
<Member Name="CUO" Datatype="Bool" />
<Member Name="CDO" Datatype="Bool" />
</Section>
</Sections>
</Member>
<Member Name="IEC_Timer_0_Instance" Datatype="TON" Version="1.0" Remanence="Retain">
<Sections>
<Section Name="Input">
<Member Name="IN" Datatype="Bool" />
<Member Name="PT" Datatype="Time" />
</Section>
<Section Name="Output">
<Member Name="Q" Datatype="Bool" />
<Member Name="ET" />
</Section>
<Section Name="InOut" />
<Section Name="Static">
<Member Name="STATE" Datatype="Byte" />
<Member Name="STIME" Datatype="Time" />
<Member Name="ATIME" Datatype="Time" />
</Section>
</Sections>
</Member>
<Member Name="Past_Entry_Sensor_1" Datatype="Bool" Remanence="Retain" />
<Member Name="Past_Start" Datatype="Bool" Remanence="Retain" />
<Member Name="Past_Exit_Sensor_1" Datatype="Bool" Remanence="Retain" />
<Member Name="Timer_Transition" Datatype="Bool" Remanence="Retain" />
<Member Name="Timer_Loading" Datatype="Bool" Remanence="Retain" />
<Member Name="IEC_Timer_0_Instance_1" Datatype="TON" Version="1.0" Remanence="Retain">
<Sections>
<Section Name="Input">
<Member Name="IN" Datatype="Bool" />
<Member Name="PT" Datatype="Time" />
</Section>

```

```

    <Section Name="Output">
        <Member Name="Q" Datatype="Bool" />
        <Member Name="ET" Datatype="Time" />
    </Section>
    <Section Name="InOut" />
    <Section Name="Static">
        <Member Name="STATE" Datatype="Byte" />
        <Member Name="STIME" Datatype="Time" />
        <Member Name="ATIME" Datatype="Time" />
    </Section>
</Sections>
</Member>
</Section>
</Sections></Interface>
<InterfaceModifiedDate ReadOnly="true">2018-04-11T09:05:54.2209102Z</InterfaceModifiedDate>
<IsConsistent ReadOnly="true">true</IsConsistent>
<IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
<MemoryLayout ReadOnly="true">Standard</MemoryLayout>
<ModifiedDate ReadOnly="true">2018-04-13T12:33:39.0349225Z</ModifiedDate>
<Name>Entry RC 11.0101</Name>
<Number>1</Number>
<ParameterModified ReadOnly="true">2018-04-11T09:05:54.2209102Z</ParameterModified>
<ProgrammingLanguage>DB</ProgrammingLanguage>
<StructureModified ReadOnly="true">2018-04-11T09:05:54.2209102Z</StructureModified>
<UDABlockProperties ReadOnly="true" />
<UDAEnableTagReadback ReadOnly="true">false</UDAEnableTagReadback>
</AttributeList>
<ObjectList>
    <MultilingualText ID="1" CompositionName="Comment">
        <ObjectList>
            <MultilingualTextItem ID="2" CompositionName="Items">
                <AttributeList>
                    <Culture>en-US</Culture>
                </AttributeList>
                <Text />
            </MultilingualTextItem>
        </ObjectList>
    </MultilingualText>
    <MultilingualText ID="3" CompositionName="Title">
        <ObjectList>
            <MultilingualTextItem ID="4" CompositionName="Items">
                <AttributeList>
                    <Culture>en-US</Culture>
                </AttributeList>
                <Text />
            </MultilingualTextItem>
        </ObjectList>
    </MultilingualText>
</ObjectList>
</SW.Blocks.InstanceDB>
</Document>

```

Conv_i (Instance DB)

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
    <Engineering version="V14 SP1" />
    <DocumentInfo>
        <Created>2018-04-19T13:21:13.9087227Z</Created>
        <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
        <InstalledProducts>
            </InstalledProducts>
        </DocumentInfo>
        <SW.Blocks.InstanceDB ID="0">
            <AttributeList>
                <AutoNumber>true</AutoNumber>
                <CodeModifiedDate ReadOnly="true">2018-04-13T12:33:38.8079095Z</CodeModifiedDate>
                <CompileDate ReadOnly="true">2018-04-13T12:33:38.81591Z</CompileDate>
                <CreationDate ReadOnly="true">2018-04-11T10:10:54.0742164Z</CreationDate>
                <HeaderAuthor />
                <HeaderFamily />
                <HeaderName />
                <HeaderVersion>0.1</HeaderVersion>
                <InstanceOfName>Conv_i</InstanceOfName>
                <InstanceOfNumber ReadOnly="true">2</InstanceOfNumber>
                <InstanceOfType ReadOnly="true">FB</InstanceOfType>
                <Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
                    <Section Name="Input">
                        <Member Name="Exit_Sensor_i_1" Datatype="Bool" Remanence="Retain" />
                        <Member Name="Exit_Sensor_i" Datatype="Bool" Remanence="Retain" />
                        <Member Name="Start" Datatype="Bool" Remanence="Retain" />
                        <Member Name="Reset" Datatype="Bool" Remanence="Retain" />
                        <Member Name="Stop" Datatype="Bool" Remanence="Retain" />
                        <Member Name="Auto" Datatype="Bool" Remanence="Retain" />
                    </Section>
                    <Section Name="Output">
                        <Member Name="Conv_i" Datatype="Bool" Remanence="Retain" />
                    </Section>
                </Sections>
            </AttributeList>
        </SW.Blocks.InstanceDB>
    </Document>

```

```

<Section Name="InOut">
  <Member Name="Transition_Time" Datatype="Time" Remanence="Retain" />
  <Member Name="CTN_Next_Conv" Datatype="Int" Remanence="Retain" />
</Section>
<Section Name="Static">
  <Member Name="E10" Datatype="Bool" Remanence="Retain" />
  <Member Name="E11" Datatype="Bool" Remanence="Retain" />
  <Member Name="E12" Datatype="Bool" Remanence="Retain" />
  <Member Name="E13" Datatype="Bool" Remanence="Retain" />
  <Member Name="E14" Datatype="Bool" Remanence="Retain" />
  <Member Name="E15" Datatype="Bool" Remanence="Retain" />
  <Member Name="Timer_Transition" Datatype="Bool" Remanence="Retain" />
  <Member Name="IEC_Counter_0_Instance" Datatype="CTUD" Version="1.0" Remanence="Retain">
    <Sections>
      <Section Name="Input">
        <Member Name="CU" Datatype="Bool" />
        <Member Name="CD" Datatype="Bool" />
        <Member Name="R" Datatype="Bool" />
        <Member Name="LOAD" Datatype="Bool" />
        <Member Name="PV" Datatype="Int" />
      </Section>
      <Section Name="Output">
        <Member Name="QU" Datatype="Bool" />
        <Member Name="QD" Datatype="Bool" />
        <Member Name="CV" Datatype="Int" />
      </Section>
      <Section Name="InOut" />
      <Section Name="Static">
        <Member Name="CUO" Datatype="Bool" />
        <Member Name="CDO" Datatype="Bool" />
      </Section>
    </Sections>
  </Member>
  <Member Name="Past_Exit_Sensor_i" Datatype="Bool" Remanence="Retain" />
  <Member Name="Past_Start" Datatype="Bool" Remanence="Retain" />
  <Member Name="IEC_Timer_0_Instance_1" Datatype="TON" Version="1.0" Remanence="Retain">
    <Sections>
      <Section Name="Input">
        <Member Name="IN" Datatype="Bool" />
        <Member Name="PT" Datatype="Time" />
      </Section>
      <Section Name="Output">
        <Member Name="Q" Datatype="Bool" />
        <Member Name="ET" Datatype="Time" />
      </Section>
      <Section Name="InOut" />
      <Section Name="Static">
        <Member Name="STATE" Datatype="Byte" />
        <Member Name="STIME" Datatype="Time" />
        <Member Name="ATIME" Datatype="Time" />
      </Section>
    </Sections>
  </Member>
</Section>
</Sections></Interface>
  <InterfaceModifiedDate ReadOnly="true">2018-04-11T10:03:50.7126997Z</InterfaceModifiedDate>
  <IsConsistent ReadOnly="true">true</IsConsistent>
  <IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
  <MemoryLayout ReadOnly="true">Standard</MemoryLayout>
  <ModifiedDate ReadOnly="true">2018-04-13T12:33:38.8079095Z</ModifiedDate>
  <Name>Entry RC 11.0202</Name>
  <Number>5</Number>
  <ParameterModified ReadOnly="true">2018-04-11T10:03:35.9021715Z</ParameterModified>
  <ProgrammingLanguage>DB</ProgrammingLanguage>
  <StructureModified ReadOnly="true">2018-04-11T10:03:50.7126997Z</StructureModified>
  <UDABlockProperties ReadOnly="true" />
  <UDAEnableTagReadback ReadOnly="true">false</UDAEnableTagReadback>
</AttributeList>
<ObjectList>
  <MultilingualText ID="1" CompositionName="Comment">
    <ObjectList>
      <MultilingualTextItem ID="2" CompositionName="Items">
        <AttributeList>
          <Culture>en-US</Culture>
          <Text />
        </AttributeList>
      </MultilingualTextItem>
    </ObjectList>
  </MultilingualText>
  <MultilingualText ID="3" CompositionName="Title">
    <ObjectList>
      <MultilingualTextItem ID="4" CompositionName="Items">
        <AttributeList>
          <Culture>en-US</Culture>
          <Text />
        </AttributeList>
      </MultilingualTextItem>
    </ObjectList>
  </MultilingualText>

```

```

</ObjectList>
</SW.Blocks.InstanceDB>
</Document>

```

Conv_n (Instance DB)

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1" />
  <DocumentInfo>
    <Created>2018-04-19T13:21:13.8407227Z</Created>
    <ExportSetting>WithDefaults, WithReadOnly</ExportSetting>
    <InstalledProducts>
    </InstalledProducts>
  </DocumentInfo>
  <SW.Blocks.InstanceDB ID="0">
    <AttributeList>
      <AutoNumber>true</AutoNumber>
      <CodeModifiedDate ReadOnly="true">2018-04-13T12:33:38.7319051Z</CodeModifiedDate>
      <CompileDate ReadOnly="true">2018-04-13T12:33:38.7429058Z</CompileDate>
      <CreationDate ReadOnly="true">2018-04-11T09:56:54.6558074Z</CreationDate>
      <HeaderAuthor />
      <HeaderFamily />
      <HeaderName />
      <HeaderVersion>0.1</HeaderVersion>
      <InstanceOfName>Conv_n</InstanceOfName>
      <InstanceOfNumber ReadOnly="true">3</InstanceOfNumber>
      <InstanceOfType ReadOnly="true">FB</InstanceOfType>
      <Interface><Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
        <Section Name="Input">
          <Member Name="Exit_Sensor_n_1" Datatype="Bool" Remanence="Retain" />
          <Member Name="Exit_Sensor_n" Datatype="Bool" Remanence="Retain" />
          <Member Name="Start" Datatype="Bool" Remanence="Retain" />
          <Member Name="Reset" Datatype="Bool" Remanence="Retain" />
          <Member Name="Stop" Datatype="Bool" Remanence="Retain" />
          <Member Name="Auto" Datatype="Bool" Remanence="Retain" />
        </Section>
        <Section Name="Output">
          <Member Name="Conv_n" Datatype="Bool" Remanence="Retain" />
        </Section>
        <Section Name="InOut">
          <Member Name="Unloading_Time" Datatype="Time" Remanence="Retain" />
        </Section>
        <Section Name="Static">
          <Member Name="E20" Datatype="Bool" Remanence="Retain" />
          <Member Name="E21" Datatype="Bool" Remanence="Retain" />
          <Member Name="E22" Datatype="Bool" Remanence="Retain" />
          <Member Name="E23" Datatype="Bool" Remanence="Retain" />
          <Member Name="E24" Datatype="Bool" Remanence="Retain" />
          <Member Name="Past_Exit_Sensor_n" Datatype="Bool" Remanence="Retain" />
          <Member Name="Past_Start" Datatype="Bool" Remanence="Retain" />
          <Member Name="IEC_Counter_0_Instance" Datatype="CTUD" Version="1.0" Remanence="Retain">
            <Sections>
              <Section Name="Input">
                <Member Name="CU" Datatype="Bool" />
                <Member Name="CD" Datatype="Bool" />
                <Member Name="R" Datatype="Bool" />
                <Member Name="LOAD" Datatype="Bool" />
                <Member Name="PV" Datatype="Int" />
              </Section>
              <Section Name="Output">
                <Member Name="QU" Datatype="Bool" />
                <Member Name="QD" Datatype="Bool" />
                <Member Name="CV" Datatype="Int" />
              </Section>
              <Section Name="InOut" />
              <Section Name="Static">
                <Member Name="CUO" Datatype="Bool" />
                <Member Name="CDO" Datatype="Bool" />
              </Section>
            </Sections>
          </Member>
          <Member Name="IEC_Timer_0_Instance_1" Datatype="TON" Version="1.0" Remanence="Retain">
            <Sections>
              <Section Name="Input">
                <Member Name="IN" Datatype="Bool" />
                <Member Name="PT" Datatype="Time" />
              </Section>
              <Section Name="Output">
                <Member Name="Q" Datatype="Bool" />
                <Member Name="ET" Datatype="Time" />
              </Section>
              <Section Name="InOut" />
              <Section Name="Static">
                <Member Name="STATE" Datatype="Byte" />
                <Member Name="STIME" Datatype="Time" />
              </Section>
            </Sections>
          </Member>

```

```

        <Member Name="ATIME" Datatype="Time" />
    </Section>
</Sections>
</Member>
<Member Name="Timer_Unloading" Datatype="Bool" Remanence="Retain" />
</Section>
</Sections></Interface>
    <InterfaceModifiedDate ReadOnly="true">2018-04-11T10:57:37.1695896Z</InterfaceModifiedDate>
    <IsConsistent ReadOnly="true">true</IsConsistent>
    <IsKnowHowProtected ReadOnly="true">false</IsKnowHowProtected>
    <MemoryLayout ReadOnly="true">Standard</MemoryLayout>
    <ModifiedDate ReadOnly="true">2018-04-13T12:33:38.7319051Z</ModifiedDate>
    <Name>Entry RC 11.0102</Name>
    <Number>3</Number>
    <ParameterModified ReadOnly="true">2018-04-11T10:57:37.1695896Z</ParameterModified>
    <ProgrammingLanguage>DB</ProgrammingLanguage>
    <StructureModified ReadOnly="true">2018-04-11T10:57:37.1695896Z</StructureModified>
    <UDABlockProperties ReadOnly="true" />
    <UDAEnableTagReadback ReadOnly="true">false</UDAEnableTagReadback>
</AttributeList>
<ObjectList>
    <MultilingualText ID="1" CompositionName="Comment">
        <ObjectList>
            <MultilingualTextItem ID="2" CompositionName="Items">
                <AttributeList>
                    <Culture>en-US</Culture>
                </AttributeList>
            </MultilingualTextItem>
        </ObjectList>
    </MultilingualText>
    <MultilingualText ID="3" CompositionName="Title">
        <ObjectList>
            <MultilingualTextItem ID="4" CompositionName="Items">
                <AttributeList>
                    <Culture>en-US</Culture>
                </AttributeList>
            </MultilingualTextItem>
        </ObjectList>
    </MultilingualText>
</ObjectList>
</SW.Blocks.InstanceDB>
</Document>

```

XML Schemas do TIA Portal utilizados na aplicação desenvolvida

SW.InterfaceSections_v2

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:include schemaLocation="SW.Common.xsd"/>
  <xs:simpleType name="Accessibility_TE">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Public"/>
      <xs:enumeration value="Internal"/>
      <xs:enumeration value="Protected"/>
      <xs:enumeration value="Private"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="IndexPath_TP">
    <xs:restriction base="xs:string">
      <xs:pattern value="-?\d+(, -?\d+)*((-?\d+(, -?\d+)*))?" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="Member" type="Member_T"/>
  <xs:complexType name="Member_T">
    <xs:sequence>
      <xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="Member"/>
        <xs:element ref="Sections"/>
        <xs:element ref="StartValue"/>
        <xs:element ref="Comment"/>
        <xs:element ref="Subelement"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="Name" type="xs:string" use="required"/>
    <xs:attribute name="Datatype" type="SimaticType_TE" use="required"/>
    <xs:attribute name="Version" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>The version of the library type to use. Previous to this, the version was written
inside the Datatype attribute itself, like "dtl:v1.0". Now, this is written in two separate attributes, to mitigate problems with weird names
("dtl:v1.0" could be a UDT name!).</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Remanence" type="Remanence_TE" default="NonRetain"/>
    <xs:attribute name="Accessibility" type="Accessibility_TE" default="Public"/>
    <xs:attribute name="Informative" type="xs:boolean" default="false"/>
  </xs:complexType>
  <xs:simpleType name="MemberAttributes_TE">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CodeReadOnly">
        <xs:annotation>
          <xs:documentation>Write acces only inside function</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="At">
        <xs:annotation>
          <xs:documentation>string: Member shares offset with another member in this
structure</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="SetPoint">
        <xs:annotation>
          <xs:documentation>boolean: Member can be synchronized with work
memory</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="UserVisible">
        <xs:annotation>
          <xs:documentation>boolean: Editor does not show the member</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="UserReadOnly">
        <xs:annotation>
          <xs:documentation>boolean: User cannot change member name</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="UserDeletable">
        <xs:annotation>
          <xs:documentation>boolean: Editor does not allow to delete the
member</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="HmiAccessible">
        <xs:annotation>
          <xs:documentation>boolean: No HMI access, no structure item</xs:documentation>
        </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="HmiVisible">
        <xs:annotation>
```

```

        <xs:documentation>boolean: Filter to reduce the number of members shown in the first
place</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="Offset">
        <xs:annotation>
            <xs:documentation>integer: </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="PaddedSize">
        <xs:annotation>
            <xs:documentation>integer: </xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="HiddenAssignment">
        <xs:annotation>
            <xs:documentation>boolean: Hide assignment at call if matches with
PredefinedAssignment</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="PredefinedAssignment">
        <xs:annotation>
            <xs:documentation>string: Input for the paramter used when call is
placed</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="ReadOnlyAssignment">
        <xs:annotation>
            <xs:documentation>boolean: The user cannot change the predefined assignment at the
call</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="Remanence_TE">
    <xs:restriction base="xs:string">
        <xs:enumeration value="SetInIDB"/>
        <xs:enumeration value="NonRetain"/>
        <xs:enumeration value="Retain"/>
    </xs:restriction>
</xs:simpleType>
<xs:element name="Section" type="Section_T"/>
<xs:complexType name="Section_T">
    <xs:sequence>
        <xs:element ref="Sections" minOccurs="0" maxOccurs="1"/>
        <xs:annotation>
            <xs:documentation>Base Class</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element ref="Member" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Name" type="SectionName_TE" use="required"/>
</xs:complexType>
<xs:element name="Sections" type="Sections_T"/>
<xs:complexType name="Sections_T">
    <xs:sequence>
        <xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="Section" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Datatype" type="SimaticType_TE"/>
    <xs:attribute name="Version" type="xs:string" use="optional"/>
</xs:complexType>
<xs:element name="StartValue" type="StartValue_T"/>
<xs:complexType name="StartValue_T">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="ConstantName" type="SimaticName_TP"/>
            <xs:attribute name="IsBulkValue" type="xs:boolean" default="false"/>
            <xs:attribute name="Informative" type="xs:boolean" default="false"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:element name="Subelement" type="Subelement_T"/>
<xs:complexType name="Subelement_T">
    <xs:sequence>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="StartValue"/>
            <xs:element ref="Comment"/>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="Path" type="IndexPath_TP"/>
</xs:complexType>
</xs:schema>

```

SW.PlcBlocks.LADFBDD

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="SW.PlcBlocks.CompileUnitCommon.xsd"/>
  <xs:element name="Powerrail" type="Powerrail_T"/>
  <xs:complexType name="Powerrail_T"/>
  <xs:element name="Openbranch" type="Openbranch_T"/>
  <xs:complexType name="Openbranch_T"/>
  <xs:element name="OpenCon" type="OpenCon_T"/>
  <xs:complexType name="OpenCon_T">
    <xs:attribute name="UId" type="xs:int" use="required"/>
  </xs:complexType>
  <xs:element name="Call" type="Call_T"/>
  <xs:complexType name="Call_T">
    <xs:sequence>
      <xs:element ref="CallInfo"/>
      <xs:group ref="PartSequence_G"/>
    </xs:sequence>
    <xs:attributeGroup ref="PartAttribute_G"/>
  </xs:complexType>
  <xs:complexType name="Equation_T">
    <xs:simpleContent>
      <xs:extension base="xs:string"/>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="FlgNet" type="FlgNet_T"/>
  <xs:complexType name="FlgNet_T">
    <xs:sequence>
      <xs:element ref="Labels" minOccurs="0"/>
      <xs:element ref="Parts"/>
      <xs:element ref="Wires" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="GateName_TE">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:element name="IdentCon" type="IdentCon_T"/>
  <xs:complexType name="IdentCon_T">
    <xs:attribute name="UId" type="xs:int" use="required"/>
  </xs:complexType>
  <xs:element name="Invisible" type="Invisible_T">
    <xs:annotation>
      <xs:documentation>The invisible pins of this part.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="Invisible_T">
    <xs:attribute name="Name" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>The name of the invisible pin.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:element name="Labels" type="Labels_T"/>
  <xs:complexType name="Labels_T">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="LabelDeclaration"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Neg_T">
    <xs:attribute name="Name" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>The name of the negated pin.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:element name="Negated" type="Neg_T">
    <xs:annotation>
      <xs:documentation>The negated pins of this part.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="AutomaticTyped_T">
    <xs:attribute name="Name" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>The name of the automatic chosen template parameter. Not for
InstructionRef</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
  <xs:element name="AutomaticTyped" type="AutomaticTyped_T"/>
  <xs:element name="Part" type="Part_T"/>
  <xs:element name="Equation" type="Equation_T"/>
  <xs:complexType name="Part_T">
    <xs:sequence>
      <xs:choice minOccurs="0">
        <xs:element ref="Equation" minOccurs="0">
          <xs:annotation>

```

```

Calculate box.</xs:documentation>
</xs:annotation>
</xs:element>
<xs:element ref="Instance" minOccurs="0"/>
</xs:choice>
<xs:group ref="PartSequence_G"/>
</xs:sequence>
<xs:attributeGroup ref="PartAttribute_G"/>
<xs:attribute name="Name" use="required">
  <xs:simpleType>
    <xs:restriction base="SimaticName_TP">
      <xs:minLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="Version" type="VersionString_TP"/>
<xs:attribute name="DisabledENO" type="xs:boolean" default="false"/>
</xs:complexType>
<xs:attributeGroup name="PartAttribute_G">
  <xs:attribute name="UId" type="xs:int" use="required"/>
</xs:attributeGroup>
<xs:element name="NameCon" type="NameCon_T"/>
<xs:complexType name="NameCon_T">
  <xs:attribute name="UId" type="xs:int" use="required"/>
  <xs:attribute name="Name" type="PinName_TE" use="required"/>
</xs:complexType>
<xs:element name="Parts" type="Parts_T"/>
<xs:complexType name="Parts_T">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="Access"/>
    <xs:element ref="Part"/>
    <xs:element ref="Call"/>
  </xs:choice>
</xs:complexType>
<xs:group name="PartSequence_G">
  <xs:sequence>
    <xs:element ref="TemplateValue" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="AutomaticTyped" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Invisible" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Negated" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Comment" minOccurs="0"/>
  </xs:sequence>
</xs:group>
<xs:simpleType name="PinName_TE">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:element name="Wire" type="Wire_T"/>
<xs:complexType name="Wire_T">
  <xs:choice maxOccurs="unbounded">
    <xs:element ref="Powerrail"/>
    <xs:element ref="NameCon"/>
    <xs:element ref="IdentCon"/>
    <xs:element ref="Openbranch"/>
    <xs:element ref="OpenCon"/>
  </xs:choice>
  <xs:attribute name="UId" type="xs:int" use="required"/>
</xs:complexType>
<xs:element name="Wires" type="Wires_T"/>
<xs:complexType name="Wires_T">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Wire"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

SW.PlcBlocks.CompileUnitCommon

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="SW.PlcBlocks.Access.xsd"/>
  <xs:element name="LabelDeclaration" type="LabelDeclaration_T"/>
  <xs:complexType name="LabelDeclaration_T">
    <xs:sequence>
      <xs:element ref="IntegerAttribute" minOccurs="0">
        <xs:annotation>
          <xs:documentation>for NumBLs. NumBLs is informative</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element ref="Label"/>
    </xs:sequence>
  </xs:complexType>
  <xs:group ref="Comment_G" minOccurs="0"/>

```

```

        <xs:sequence minOccurs="0">
            <xs:element ref="Token">
                <xs:annotation>
                    <xs:documentation>the COLON; only if separated</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:group ref="Comment_G" minOccurs="0"/>
        </xs:sequence>
    </xs:sequence>
    <xs:attribute name="UId" type="xs:int">
        <xs:annotation>
            <xs:documentation>Not allowed in STL</xs:documentation>
        </xs:annotation>
    </xs:attribute>
</xs:complexType>
</xs:schema>

```

SW.PlcBlocks.Access

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="SW.Common.xsd"/>
    <xs:element name="Access" type="Access_T"/>
    <xs:complexType name="Access_T">
        <xs:sequence>
            <xs:element ref="IntegerAttribute" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>for NumBLs. NumBLs is informative. Not for LAD/FBD.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:choice>
                <xs:element ref="Label"/>
                <xs:element ref="Constant"/>
                <xs:element ref="CallInfo">
                    <xs:annotation>
                        <xs:documentation>call of a user block. Not in Graph ActionList.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="Instruction">
                    <xs:annotation>
                        <xs:documentation>call of an instruction. Not for LAD/FBD, Graph ActionList.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="Indirect">
                    <xs:annotation>
                        <xs:documentation>STL specific</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="Statusword"/>
                <xs:element ref="Expression">
                    <xs:annotation>
                        <xs:documentation>SCL specific</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="Symbol"/>
                <xs:element ref="Address">
                    <xs:annotation>
                        <xs:documentation>for absolute addresses</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:choice>
            <xs:group ref="Comment_G" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="Scope" use="required">
            <xs:simpleType>
                <xs:restriction base="Scope_TE"/>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="UId" type="xs:int" use="optional">
            <xs:annotation>
                <xs:documentation>Not allowed in STL</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="Address" type="Address_T"/>
    <xs:complexType name="Address_T">
        <xs:attribute name="Area" type="Area_TE" use="required"/>
        <xs:attribute name="Type" type="SimaticName_TP" use="optional"/>
        <xs:attribute name="BlockNumber" type="xs:int" use="optional">
            <xs:annotation>
                <xs:documentation>for DB access</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>

```

```

<xs:attribute name="BitOffset" type="xs:int" use="optional">
  <xs:annotation>
    <xs:documentation>In general it is Byte * 8 + Bit. But if it is used for addressing a DB we will find the number of the DB here (e.g.
"DB12" ->12).</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Informative" type="xs:boolean" default="false">
  <xs:annotation>
    <xs:documentation>if true, the import unnoted it</xs:documentation>
  </xs:annotation>
</xs:attribute>
</xs:complexType>
<xs:element name="AbsoluteOffset" type="AbsoluteOffset_T"/>
<xs:complexType name="AbsoluteOffset_T">
  <xs:attribute name="BitOffset" type="xs:int" use="required">
    <xs:annotation>
      <xs:documentation>Byte * 8 + Bit</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="Type" type="xs:string" use="required"/>
</xs:complexType>
<xs:simpleType name="Area_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="Periphery">
      <xs:annotation>
        <xs:documentation>only used for incomplete pointer</xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="PeripheryInput"/>
    <xs:enumeration value="PeripheryOutput"/>
    <xs:enumeration value="Input"/>
    <xs:enumeration value="Output"/>
    <xs:enumeration value="Memory"/>
    <xs:enumeration value="FB"/>
    <xs:enumeration value="FC"/>
    <xs:enumeration value="DB">
      <xs:annotation>
        <xs:documentation>partly qualified access with DB register</xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="DI">
      <xs:annotation>
        <xs:documentation>partly qualified access with DI register</xs:documentation>
      </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="Timer"/>
    <xs:enumeration value="Counter"/>
    <xs:enumeration value="Local">
      <xs:annotation>
        <xs:documentation>Classic Local Stack</xs:documentation>
      </xs:annotation>
    </xs:enumeration>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="BlockType_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="DB"/>
    <xs:enumeration value="FB"/>
    <xs:enumeration value="FC"/>
    <xs:enumeration value="OB"/>
    <xs:enumeration value="UDT"/>
    <xs:enumeration value="FBT"/>
    <xs:enumeration value="FCT"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="CallInfo">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="CallInfo_T"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:complexType name="CallInfo_T">
  <xs:annotation>
    <xs:documentation>Not for LAD/FBD. </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element ref="IntegerAttribute" minOccurs="0">
      <xs:annotation>
        <xs:documentation>for BlockNumber. BlockNumber is informative.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="DateAttribute" minOccurs="0">
      <xs:annotation>
        <xs:documentation>for ParameterModifiedTS. ParameterModifiedTS is informative</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:group ref="Comment_G" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

```

```

<xs:sequence minOccurs="0">
  <xs:element ref="Token">
    <xs:annotation>
      <xs:documentation>the COMMA, only if separated. Not for LAD/FBD</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:group ref="Comment_G" minOccurs="0"/>
</xs:sequence>
<xs:element ref="Instance" minOccurs="0"/>
<xs:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Name" type="SimaticName_TP" use="optional"/>
<xs:attribute name="BlockType" type="BlockType_TE" use="required"/>
</xs:complexType>
<xs:element name="Component" type="Component_T"/>
<xs:complexType name="Component_T">
  <xs:sequence minOccurs="0">
    <xs:annotation>
      <xs:documentation>For the indices of an array</xs:documentation>
    </xs:annotation>
    <xs:element ref="Access" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Name" type="xs:string" use="required"/>
  <xs:attribute name="SliceAccessModifier" type="SliceAccessModifier_TP" default="undef"/>
  <xs:attribute name="SimpleAccessModifier" type="SimpleAccessModifier_TP" default="None"/>
</xs:complexType>
<xs:element name="Constant" type="Constant_T"/>
<xs:complexType name="Constant_T">
  <xs:sequence>
    <xs:element ref="ConstantType" minOccurs="0"/>
    <xs:element ref="ConstantValue" minOccurs="0"/>
    <xs:element ref="StringAttribute" minOccurs="0" maxOccurs="2">
      <xs:annotation>
        <xs:documentation>for Format and FormatFlags. They are informative..</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="Name" type="SimaticName_TP"/>
</xs:complexType>
<xs:element name="ConstantValue">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="ConstantValue_T">
        <xs:attribute name="Informative" type="xs:boolean" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:complexType name="ConstantValue_T">
  <xs:simpleContent>
    <xs:extension base="xs:string"/>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="Expression" type="Expression_T"/>
<xs:complexType name="Expression_T">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Access"/>
      <xs:element ref="Token"/>
    </xs:choice>
    <xs:group ref="Comment_G" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="UId" type="xs:int"/>
</xs:complexType>
<xs:simpleType name="Format_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Real"/>
    <xs:enumeration value="Bin"/>
    <xs:enumeration value="DecSigned"/>
    <xs:enumeration value="DecUnsigned"/>
    <xs:enumeration value="Pointer"/>
    <xs:enumeration value="CharSequence"/>
    <xs:enumeration value="DecSequence"/>
    <xs:enumeration value="Hex"/>
    <xs:enumeration value="S5Count"/>
    <xs:enumeration value="Time"/>
    <xs:enumeration value="Date"/>
    <xs:enumeration value="TimeOfDay"/>
    <xs:enumeration value="S5Time"/>
    <xs:enumeration value="Bool"/>
    <xs:enumeration value="Oct"/>
    <xs:enumeration value="Bcd"/>
    <xs:enumeration value="DateAndTime"/>
    <xs:enumeration value="String"/>
    <xs:enumeration value="Any"/>
    <xs:enumeration value="Number"/>
    <xs:enumeration value="Char"/>
    <xs:enumeration value="HexSequence"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormatFlags_TE">

```

```

<xs:restriction base="xs:string">
  <xs:pattern value="None"/>
  <xs:pattern value="((Lower|Format|Size|Under|Exp|TypeQualifier)(,|s*)?)*/>
</xs:restriction>
</xs:simpleType>
<xs:element name="Indirect" type="Indirect_T"/>
<xs:complexType name="Indirect_T">
  <xs:sequence minOccurs="0">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="Token"/>
      <xs:group ref="Comment_G"/>
    </xs:choice>
    <xs:element ref="Access" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Width" type="Width_TE" use="required"/>
  <xs:attribute name="Area" type="Area_TE" use="optional"/>
  <xs:attribute name="Register" type="Register_TE" use="optional"/>
  <xs:attribute name="BitOffset" use="optional"/>
</xs:complexType>
<xs:element name="Instance" type="Instance_T"/>
<xs:element name="Instruction" type="Instruction_T"/>
<xs:complexType name="Instruction_T">
  <xs:sequence>
    <xs:group ref="Comment_G" minOccurs="0"/>
    <xs:sequence minOccurs="0">
      <xs:element ref="Token">
        <xs:annotation>
          <xs:documentation>the COMMA, only if separated</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:group ref="Comment_G" minOccurs="0"/>
    </xs:sequence>
    <xs:element ref="TemplateValue" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Instance" minOccurs="0"/>
    <xs:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="Name" use="required">
    <xs:simpleType>
      <xs:restriction base="SimaticName_TP">
        <xs:minLength value="1"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="Version" type="VersionString_TP"/>
</xs:complexType>
<xs:simpleType name="InterfaceFlags_TP">
  <xs:restriction base="xs:string">
    <xs:pattern value="None"/>
    <xs:pattern value="((Mandatory|S7_Visible)(,|s*)?)*/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="Label" type="Label_T"/>
<xs:complexType name="Label_T">
  <xs:attribute name="Name" type="SimaticName_TP" use="required"/>
</xs:complexType>
<xs:element name="Parameter" type="Parameter_T"/>
<xs:complexType name="Parameter_T">
  <xs:sequence>
    <xs:element ref="IntegerAttribute" minOccurs="0">
      <xs:annotation>
        <xs:documentation>for NumBLs. NumBLs is informative</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="StringAttribute" minOccurs="0">
      <xs:annotation>
        <xs:documentation>for InterfaceFlags. InterfaceFlags is informative</xs:documentation>
        <xs:documentation>The type of the value should be InterfaceFlags_TP</xs:documentation>
        <xs:documentation>The default value is "S7_Visible"</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:group ref="Comment_G" minOccurs="0"/>
    <xs:group ref="Access_G" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Name" type="SimaticName_TP" use="required"/>
  <xs:attribute name="Section" type="SectionName_TE" use="optional"/>
  <xs:attribute name="Type" type="SimaticType_TE"/>
  <xs:attribute name="TemplateReference" type="xs:string"/>
</xs:complexType>
<xs:simpleType name="Register_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AR1"/>
    <xs:enumeration value="AR2"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Scope_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Undef">
      <xs:annotation>
        <xs:documentation>Symbols we do not know what they are</xs:documentation>
      </xs:annotation>
    </xs:enumeration>
  </xs:restriction>
</xs:simpleType>

```

```

</xs:annotation>
</xs:enumeration>
<xs:enumeration value="GlobalConstant"/>
<xs:enumeration value="LocalConstant"/>
<xs:enumeration value="GlobalVariable"/>
<xs:enumeration value="LocalVariable"/>
<xs:enumeration value="Instruction"/>
<xs:enumeration value="Label"/>
<xs:enumeration value="TypedConstant"/>
<xs:enumeration value="AddressConstant"/>
<xs:enumeration value="LiteralConstant"/>
<xs:enumeration value="AlarmConstant"/>
<xs:enumeration value="Address"/>
<xs:enumeration value="Statusword"/>
<xs:enumeration value="Expression"/>
<xs:enumeration value="Unnamed"/>
<xs:enumeration value="Call"/>
<xs:enumeration value="CallWithType"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="SimpleAccessModifier_TP">
  <xs:restriction base="xs:string">
    <xs:pattern value="None(((Periphery|QualityInformation)(,\\s*)?))*"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="SimpleType_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="undef"/>
    <xs:enumeration value="Bool"/>
    <xs:enumeration value="Byte"/>
    <xs:enumeration value="Char"/>
    <xs:enumeration value="Word"/>
    <xs:enumeration value="Int"/>
    <xs:enumeration value="DWord"/>
    <xs:enumeration value="DInt"/>
    <xs:enumeration value="Real"/>
    <xs:enumeration value="LReal"/>
    <xs:enumeration value="Timer"/>
    <xs:enumeration value="SSTime"/>
    <xs:enumeration value="ARef"/>
    <xs:enumeration value="Any"/>
    <xs:enumeration value="Time"/>
    <xs:enumeration value="S5Count"/>
    <xs:enumeration value="Counter"/>
    <xs:enumeration value="Block_DB"/>
    <xs:enumeration value="Block_FB"/>
    <xs:enumeration value="Block_FC"/>
    <xs:enumeration value="Block_SFB"/>
    <xs:enumeration value="Block_UDT"/>
    <xs:enumeration value="Multi_FB"/>
    <xs:enumeration value="Multi_SFB"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="SliceAccessModifier_TP">
  <xs:restriction base="xs:string">
    <xs:pattern value="([xbwdXBWD]\\d+)]undef"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="Statusword" type="Statusword_T">
  <xs:annotation>
    <xs:documentation>Only for S7-300/400/WinAC</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:complexType name="Statusword_T">
  <xs:attribute name="Combination" type="Statusword_TE" use="required"/>
</xs:complexType>
<xs:simpleType name="Statusword_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="BR"/>
    <xs:enumeration value="OV"/>
    <xs:enumeration value="OS"/>
    <xs:enumeration value="EQ"/>
    <xs:enumeration value="NE"/>
    <xs:enumeration value="GT"/>
    <xs:enumeration value="LT"/>
    <xs:enumeration value="GE"/>
    <xs:enumeration value="LE"/>
    <xs:enumeration value="UO"/>
    <xs:enumeration value="NU"/>
    <xs:enumeration value="STW"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="Symbol" type="Symbol_T"/>
<xs:complexType name="Symbol_T">
  <xs:sequence>
    <xs:sequence maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="Component"/>
        <xs:element ref="AbsoluteOffset"/>
        <xs:element ref="Token"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:annotation>

```

```

    <xs:documentation>the DOT; only if separated. Not in Graph ActionList, not in LAD/FBD.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:choice>
<xs:group ref="Comment_G" minOccurs="0"/>
</xs:sequence>
<xs:element ref="Address" minOccurs="0">
  <xs:annotation>
    <xs:documentation>additional address for a symbol. it is informative</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
<xs:attribute name="UId" type="xs:int">
  <xs:annotation>
    <xs:documentation>Not allowed in STL</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Scope" type="Scope_TE"/>
</xs:complexType>
<xs:complexType name="Instance_T">
  <xs:sequence>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="Component"/>
        <xs:element ref="AbsoluteOffset"/>
        <xs:element ref="Token">
          <xs:annotation>
            <xs:documentation>the DOT; only if separated. Not in Graph ActionList, not in LAD/FBD.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
    </xs:sequence>
    <xs:group ref="Comment_G" minOccurs="0"/>
  </xs:sequence>
  <xs:element ref="Address" minOccurs="0">
    <xs:annotation>
      <xs:documentation>additional address for a symbol. it is informative</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:sequence>
<xs:attribute name="UId" type="xs:int">
  <xs:annotation>
    <xs:documentation>Not allowed in STL</xs:documentation>
  </xs:annotation>
</xs:attribute>
<xs:attribute name="Scope" type="Scope_TE" use="required"/>
</xs:complexType>
<xs:element name="TemplateValue" type="TemplateValue_T"/>
<xs:complexType name="TemplateValue_T">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="Name" type="SimaticName_TP" use="required"/>
      <xs:attribute name="Type" type="TemplateType_TE" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="TemplateType_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Cardinality"/>
    <xs:enumeration value="Type"/>
    <xs:enumeration value="Operation"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Width_TE">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="Bit"/>
    <xs:enumeration value="Byte"/>
    <xs:enumeration value="Word"/>
    <xs:enumeration value="Offset"/>
    <xs:enumeration value="Double"/>
    <xs:enumeration value="Pointer"/>
    <xs:enumeration value="Long"/>
    <xs:enumeration value="Any"/>
    <xs:enumeration value="Block"/>
  </xs:restriction>
</xs:simpleType>
<xs:group name="Access_G">
  <xs:sequence>
    <xs:element ref="Access"/>
    <xs:group ref="Comment_G" minOccurs="0"/>
  </xs:sequence>
</xs:group>
<xs:element name="ConstantType">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Informative" type="xs:boolean" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

```

```
</xs:complexType>
</xs:element>
</xs:schema>
```

SW.Common

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="AttributeList" type="AttributList_T"/>
  <xs:complexType name="AttributList_T">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="BooleanAttribute"/>
      <xs:element ref="IntegerAttribute"/>
      <xs:element ref="RealAttribute"/>
      <xs:element ref="StringAttribute"/>
    </xs:choice>
  </xs:complexType>
  <xs:element name="BooleanAttribute" type="BooleanAttribute_T">
    <xs:annotation>
      <xs:documentation>A member attribute with a type restriction of boolean.</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="BooleanAttribute_T">
    <xs:simpleContent>
      <xs:extension base="xs:boolean">
        <xs:attribute name="Name" type="xs:string" use="required"/>
        <xs:attribute name="Informative" type="xs:boolean" default="false">
          <xs:annotation>
            <xs:documentation>Exported only with ReadOnly option, ignored during
import.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
          <xs:annotation>
            <xs:documentation>An attribute of attribute, denotes if it is defined by a
user or the system itself. In V14, if exists it is always true.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="Comment">
    <xs:annotation>
      <xs:documentation>Not allowed in STL</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="Comment_T">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element ref="Comment"/>
        <xs:element ref="LineComment"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:group name="Comment_G">
    <xs:annotation>
      <xs:documentation>LAD/FBD: Only for Parts</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element ref="Comment"/>
        <xs:element ref="MultiLanguageText"/>
      </xs:choice>
    </xs:sequence>
  </xs:group>
  <xs:complexType name="Comment_T">
    <xs:sequence minOccurs="0">
      <xs:element ref="IntegerAttribute" minOccurs="0">
        <xs:annotation>
          <xs:documentation>For NumBLs. NumBLs is the count of the blank spaces before the
actual text in the Comment. This is informative.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="MultiLanguageText"/>
      </xs:sequence>
    </xs:sequence>
    <xs:attribute name="Inserted" type="xs:boolean" default="false">
      <xs:annotation>
        <xs:documentation>Denotes if the comment is at the end of the line (using //) or inside the line
(using /* */) </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="Informative" type="xs:boolean" default="false">
      <xs:annotation>
        <xs:documentation>Exported only with ReadOnly option, ignored during
import.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
```

```

        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
    <xs:simpleType name="GUID_TP">
      <xs:restriction base="xs:string">
        <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:element name="IntegerAttribute" type="IntegerAttribute_T">
      <xs:annotation>
        <xs:documentation>A member attribute with a type restriction of integer.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:complexType name="IntegerAttribute_T">
      <xs:annotation>
        <xs:documentation>Not for LAD/FBD.</xs:documentation>
      </xs:annotation>
      <xs:simpleContent>
        <xs:extension base="xs:integer">
          <xs:attribute name="Name" type="xs:string" use="required"/>
          <xs:attribute name="Informative" type="xs:boolean" default="false">
            <xs:annotation>
              <xs:documentation>Exported only with ReadOnly option, ignored during
import.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
            <xs:annotation>
              <xs:documentation>An attribute of attribute, denotes if it is defined by a
user or the system itself. In V14, if exists it is always true.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="Lang_TP">
      <xs:restriction base="xs:string">
        <xs:pattern value="[a-zA-Z]{2}-[a-zA-Z]{2}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:element name="LineComment" type="LineComment_T">
      <xs:annotation>
        <xs:documentation>Not for LAD/FBD </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:complexType name="LineComment_T">
      <xs:sequence>
        <xs:element ref="IntegerAttribute" minOccurs="0">
          <xs:annotation>
            <xs:documentation>For NumBLs. NumBLs is the count of the blank spaces before the
actual text in the LineComment. This is informative.</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="Text">
          <xs:annotation>
            <xs:documentation>the value of the comment</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="Inserted" type="xs:boolean" default="false">
        <xs:annotation>
          <xs:documentation>Denotes if the comment is at the end of the line (using //) or inside the line
(using /* */) </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
    <xs:element name="MultiLanguageText" type="MultiLanguageText_T"/>
    <xs:complexType name="MultiLanguageText_T">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="Lang" type="Lang_TP" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <xs:element name="RealAttribute" type="RealAttribute_T">
      <xs:annotation>
        <xs:documentation>A member attribute with a type restriction of real.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:complexType name="RealAttribute_T">
      <xs:simpleContent>
        <xs:extension base="xs:double">
          <xs:attribute name="Name" type="xs:string" use="required"/>
          <xs:attribute name="Informative" type="xs:boolean" default="false">
            <xs:annotation>
              <xs:documentation>Exported only with ReadOnly option, ignored during
import.</xs:documentation>
            </xs:annotation>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:schema>

```

```

        <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
            <xs:annotation>
                <xs:documentation>An attribute of attribute, denotes if it is defined by a
user or the system itself. In V14, if exists it is always true.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:simpleType name="SectionName_TE">
    <xs:restriction base="xs:string">
        <xs:enumeration value="None"/>
        <xs:enumeration value="Input"/>
        <xs:enumeration value="Return"/>
        <xs:enumeration value="Output"/>
        <xs:enumeration value="InOut"/>
        <xs:enumeration value="Static"/>
        <xs:enumeration value="Temp"/>
        <xs:enumeration value="Constant"/>
        <xs:enumeration value="Base"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="SimaticName_TP">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="SimaticType_TE">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:element name="StringAttribute" type="StringAttribute_T">
    <xs:annotation>
        <xs:documentation>A member attribute with a type restriction of string.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:complexType name="StringAttribute_T">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="Name" type="xs:string" use="required"/>
            <xs:attribute name="Informative" type="xs:boolean" default="false">
                <xs:annotation>
                    <xs:documentation>Exported only with ReadOnly option, ignored during
import.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
                <xs:annotation>
                    <xs:documentation>An attribute of attribute, denotes if it is defined by a
user or the system itself. In V14, if exists it is always true.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:element name="DateAttribute" type="DateAttribute_T"/>
<xs:complexType name="DateAttribute_T">
    <xs:simpleContent>
        <xs:extension base="xs:dateTime">
            <xs:attribute name="Name" type="xs:string" use="required"/>
            <xs:attribute name="Informative" type="xs:boolean" default="false">
                <xs:annotation>
                    <xs:documentation>Exported only with ReadOnly option, ignored during
import.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
                <xs:annotation>
                    <xs:documentation>An attribute of attribute, denotes if it is defined by a
user or the system itself. In V14, if exists it is always true.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:element name="Text" type="Text_T"/>
<xs:complexType name="Text_T">
    <xs:simpleContent>
        <xs:extension base="xs:string"/>
    </xs:simpleContent>
</xs:complexType>
<xs:element name="Token" type="Token_T"/>
<xs:complexType name="Token_T">
    <xs:sequence minOccurs="0">
        <xs:element ref="IntegerAttribute" minOccurs="0">
            <xs:annotation>
                <xs:documentation>For NumBLs. NumBLs is the count of the blank spaces at the
start.This is informative.</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="Uid" type="xs:int">
        <xs:annotation>
            <xs:documentation>Not allowed in STL</xs:documentation>
        </xs:annotation>
    </xs:attribute>

```

```

        </xs:annotation>
        </xs:attribute>
        <xs:attribute name="Text" type="Token_TE" use="required"/>
    </xs:complexType>
    <xs:simpleType name="Token_TE">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="VersionString_TP">
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]+(.{0-9}){0,3}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="ViewInfo" type="ViewInfo_T"/>
    <xs:complexType name="ViewInfo_T">
        <xs:attribute name="Start" type="xs:boolean" use="optional"/>
        <xs:attribute name="XPos" type="xs:int" use="optional"/>
        <xs:attribute name="YPos" type="xs:int" use="optional"/>
        <xs:attribute name="Width" type="xs:int" use="optional"/>
        <xs:attribute name="Height" type="xs:int" use="optional"/>
    </xs:complexType>
</xs:schema>
```

Classes Utilizadas na Manipulação dos documentos XML

Organization Blocks

```

using System.Xml;
using System.Xml.Serialization;

namespace AppGeneratoar
{
    #region ADDED CLASSES

    public abstract class Node
    {
        [XmlElement]
        public XmlElement[] OtherElements { get; set; }

        [XmlAttribute]
        public XmlAttribute[] OtherAttributes { get; set; }

        // Classes inherit from Node
        public class Document : Node
        {
            [XmlElement(ElementName = "SW.Blocks.OB")]
            public Blocks Blocks_ { get; set; }

            public class Blocks : Node
            {
                [XmlElement("AttributeList", Namespace = "")]
                public AttributList_T AttributeList { get; set; }

                [XmlElement("ObjectList", Namespace = "")]
                public ObjectList_T ObjectList { get; set; }

                [XmlAttribute("ID")]
                public string ID { get; set; }

                [XmlRoot(Namespace = "http://www.siemens.com/automation/Openness/SW/Interface/v2")]
                public class Interface : Node
                {
                    [XmlElement("Sections", Namespace = "")]
                    public Sections_T Sections { get; set; }

                    [System.Xml.Serialization.XmlRootAttribute("ObjectList", Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1", IsNullable = false)]
                    public class ObjectList_T : Node
                    {
                        [XmlElement("SW.Blocks.CompileUnit", Namespace = "")]
                        public SWCompile_T[] SWCompile { get; set; }

                        [XmlRoot(Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1")]
                        public class SWCompile_T : Node
                        {
                            [XmlElement("AttributeList", Namespace = "")]
                            public AttributList_T AttributeList { get; set; }

                            [XmlElement("ObjectList", Namespace = "")]
                            public ObjectList_T ObjectList { get; set; }

                            [XmlAttribute("ID")]
                            public string ID { get; set; }

                            [XmlAttribute("CompositionName")]
                            public string CompName { get; set; }

                            public class NetSource_T
                            {
                                [XmlElement("FlgNet", Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1")]
                                public FlgNet_T flgNet { get; set; }

                                public class Name_T
                                {
                                    [XmlText]
                                    public string Name { get; set; }

                                    public class Number_T
                                    {
                                        [XmlText]
                                        public string Nmb { get; set; }

                                        public class ProgLang_T
                                        {
                                            [XmlText]
                                            public string Lan { get; set; }

                                            public class SType_T
                                            {
                                                [XmlText]
                                                public string Type { get; set; }

                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }

        #endregion

        #region AUTO-GENERATED CLASSES
//-----
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:4.0.30319.42000
//
// Changes to this file may cause incorrect behavior and will
// be lost if
// the code is regenerated.

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("AttributeList", Namespace = "", IsNullable = false)]
public partial class AttributList_T : Node
{
    private object[] itemsField;

    [System.Xml.Serialization.XmlElementAttribute("BooleanAttribute", typeof(BooleanAttribute_T))]
    [System.Xml.Serialization.XmlElementAttribute("IntegerAttribute", typeof(IntegerAttribute_T))]
    [System.Xml.Serialization.XmlElementAttribute("RealAttribute", typeof(RealAttribute_T))]
    [System.Xml.Serialization.XmlElementAttribute("StringAttribute", typeof(StringAttribute_T))]
    public object[] Items { get { return this.itemsField; } set { this.itemsField = value; } }

    [XmlElement("Interface")]
    public Interface Interface_ { get; set; }
    [XmlElement("NetworkSource")]
    public NetSource_T NetSource { get; set; }
    //Added 17/04
    [XmlElement("Name")]
    public Name_T Name { get; set; }
    [XmlElement("Number")]
    public Number_T Number { get; set; }
    [XmlElement("ProgrammingLanguage")]
    public ProgLang_T Lang { get; set; }
    [XmlElement("SecondaryType")]
    public SType_T Type { get; set; }

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("BooleanAttribute", Namespace = "", IsNullable = false)]
    public partial class BooleanAttribute_T
    {
        private string nameField;
        // Altered the order (first: SystemDefined, then: Informative)
        private bool systemDefinedField;
        private bool informativeField;
        private bool valueField;
        public BooleanAttribute_T() { this.informativeField = false; this.systemDefinedField = true; }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Name { get { return this.nameField; } set { this.nameField = value; } }
        // I changed this default attribute value to false and altered the order (first: SystemDefined, then: Informative)

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool SystemDefined { get { return this.systemDefinedField; } set { this.systemDefinedField = value; } }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool Informative { get { return this.informativeField; } set { this.informativeField = value; } }

        [System.Xml.Serialization.XmlTextAttribute()]
        public bool Value { get { return this.valueField; } set { this.valueField = value; } }

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        public partial class Comment_T
        {
            private IntegerAttribute_T integerAttributeField;
            private MultiLanguageText_T[] multiLanguageTextField;
            private bool insertedField;
            private bool informativeField;

            public Comment_T() { this.insertedField = false; this.informativeField = false; }
        }
    }
}

```

```

        public IntegerAttribute_T IntegerAttribute { get { return this.integerAttributeField; } set { this.integerAttributeField = value; } }

        [System.Xml.Serialization.XmlElementAttribute("MultiLanguageText")]
        public MultiLanguageText_T[] MultiLanguageText { get { return this.multiLanguageTextField; } set { this.multiLanguageTextField = value; } }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool Inserted { get { return this.insertedField; } set { this.insertedField = value; } }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool Informative { get { return this.informativeField; } set { this.informativeField = value; } }

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("IntegerAttribute", Namespace = "", IsNullable = false)]
        public partial class IntegerAttribute_T
        {
            private string nameField;
            // Altered the order (first: SystemDefined, then: Informative)
            private bool systemDefinedField;
            private bool informativeField;
            private string valueField;

            public IntegerAttribute_T() { this.informativeField = false; this.systemDefinedField = true; }

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public string Name { get { return this.nameField; } set { this.nameField = value; } }

            // I changed this default attribute value to false and altered the order (first: SystemDefined, then: Informative)
            [System.Xml.Serialization.XmlAttributeAttribute()]
            [System.ComponentModel.DefaultValueAttribute(false)]
            public bool SystemDefined { get { return this.systemDefinedField; } set { this.systemDefinedField = value; } }

            [System.Xml.Serialization.XmlAttributeAttribute()]
            [System.ComponentModel.DefaultValueAttribute(false)]
            public bool Informative { get { return this.informativeField; } set { this.informativeField = value; } }

            [System.Xml.Serialization.XmlTextAttribute(DataType = "integer")]
            public string Value { get { return this.valueField; } set { this.valueField = value; } }
        }

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("MultiLanguageText", Namespace = "", IsNullable = false)]
        public partial class MultiLanguageText_T
        {
            private string langField;
            private string valueField;

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public string Lang { get { return this.langField; } set { this.langField = value; } }

            [System.Xml.Serialization.XmlTextAttribute()]
            public string Value { get { return this.valueField; } set { this.valueField = value; } }
        }

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("RealAttribute", Namespace = "", IsNullable = false)]
        public partial class RealAttribute_T
        {
            private string nameField;
            private bool informativeField;
            private bool systemDefinedField;
            private double valueField;

            public RealAttribute_T()
            {
                this.informativeField = false;
                this.systemDefinedField = true; }

```

```

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Name { get { return this.nameField; } set { this.nameField = value; } }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool Informative { get { return this.informativeField; } set { this.informativeField = value; } }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(true)]
        public bool SystemDefined { get { return this.systemDefinedField; } set { this.systemDefinedField = value; } }

        [System.Xml.Serialization.XmlTextAttribute()]
        public double Value { get { return this.valueField; } set { this.valueField = value; } }

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("StringAttribute", Namespace = "", IsNullable = false)]
        public partial class StringAttribute_T
        {
            private string nameField;
            private bool informativeField;
            private bool systemDefinedField;
            private string valueField;

            public StringAttribute_T()
            { this.informativeField = false; this.systemDefinedField = true; }

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public string Name { get { return this.nameField; } set { this.nameField = value; } }

            [System.Xml.Serialization.XmlAttributeAttribute()]
            [System.ComponentModel.DefaultValueAttribute(false)]
            public bool Informative { get { return this.informativeField; } set { this.informativeField = value; } }

            [System.Xml.Serialization.XmlAttributeAttribute()]
            [System.ComponentModel.DefaultValueAttribute(true)]
            public bool SystemDefined { get { return this.systemDefinedField; } set { this.systemDefinedField = value; } }

            [System.Xml.Serialization.XmlTextAttribute()]
            public string Value { get { return this.valueField; } set { this.valueField = value; } }
        }

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlTypeAttribute(AnonymousType = true)]
        [System.Xml.Serialization.XmlRootAttribute(Namespace = "", IsNullable = false)]
        public partial class Comment : Comment_T
        {
            [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
            [System.SerializableAttribute()]
            [System.Diagnostics.DebuggerStepThroughAttribute()]
            [System.ComponentModel.DesignerCategoryAttribute("code")]
            [System.Xml.Serialization.XmlRootAttribute("LineComment", Namespace = "", IsNullable = false)]
            public partial class LineComment_T
            {
                private IntegerAttribute_T integerAttributeField;
                private Text_T textField;
                private bool insertedField;

                public LineComment_T()
                { this.insertedField = false; }

                public IntegerAttribute_T IntegerAttribute { get { return this.integerAttributeField; } set { this.integerAttributeField = value; } }

                public Text_T Text { get { return this.textField; } set { this.textField = value; } }

                [System.Xml.Serialization.XmlAttributeAttribute()]
                [System.ComponentModel.DefaultValueAttribute(false)]
                public bool Inserted { get { return this.insertedField; } set { this.insertedField = value; } }
            }

            [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
            [System.SerializableAttribute()]

```

```

[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Text", Namespace = "", IsNullable = false)]
public partial class Text_T
{
    private string valueField;

    [System.Xml.Serialization.XmlTextAttribute()]
    public string Value { get { return this.valueField; } set { this.valueField = value; } }

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("DateAttribute", Namespace = "", IsNullable = false)]
    public partial class DateAttribute_T
    {
        private string nameField;
        private bool informativeField;
        private bool systemDefinedField;
        private System.DateTime valueField;

        public DateAttribute_T()
        { this.informativeField = false; this.systemDefinedField = true; }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Name
        { get { return this.nameField; } set { this.nameField = value; } }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool Informative
        { get { return this.informativeField; } set { this.informativeField = value; } }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(true)]
        public bool SystemDefined
        { get { return this.systemDefinedField; } set { this.systemDefinedField = value; } }

        [System.Xml.Serialization.XmlTextAttribute()]
        public System.DateTime Value
        { get { return this.valueField; } set { this.valueField = value; } }

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("Token", Namespace = "", IsNullable = false)]
        public partial class Token_T
        {
            private IntegerAttribute_T integerAttributeField;
            private int uIdField;
            private bool uIdFieldSpecified;
            private string textField;

            public IntegerAttribute_T IntegerAttribute
            { get { return this.integerAttributeField; } set { this.integerAttributeField = value; } }

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public int UId
            { get { return this.uIdField; } set { this.uIdField = value; } }

            [System.Xml.Serialization.XmlIgnoreAttribute()]
            public bool UIdSpecified
            { get { return this.uIdFieldSpecified; } set { this.uIdFieldSpecified = value; } }

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public string Text
            { get { return this.textField; } set { this.textField = value; } }

            [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
            [System.SerializableAttribute()]
            [System.Diagnostics.DebuggerStepThroughAttribute()]
            [System.ComponentModel.DesignerCategoryAttribute("code")]
            [System.Xml.Serialization.XmlRootAttribute("ViewInfo", Namespace = "", IsNullable = false)]
            public partial class ViewInfo_T
            {
                private bool startField;
                private bool startFieldSpecified;
                private int xPosField;
                private bool xPosFieldSpecified;
                private int yPosField;
                private bool yPosFieldSpecified;
                private int widthField;
                private bool widthFieldSpecified;
                private int heightField;
                private bool heightFieldSpecified;

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public bool Start
                { get { return this.startField; } set { this.startField = value; } }

                [System.Xml.Serialization.XmlIgnoreAttribute()]
                public bool StartFieldSpecified
                { get { return this.startFieldSpecified; } set { this.startFieldSpecified = value; } }

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public int XPos
                { get { return this.xPosField; } set { this.xPosField = value; } }

                [System.Xml.Serialization.XmlIgnoreAttribute()]
                public bool XPosSpecified
                { get { return this.xPosFieldSpecified; } set { this.xPosFieldSpecified = value; } }

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public int YPos
                { get { return this.yPosField; } set { this.yPosField = value; } }

                [System.Xml.Serialization.XmlIgnoreAttribute()]
                public bool YPosSpecified
                { get { return this.yPosFieldSpecified; } set { this.yPosFieldSpecified = value; } }

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public int Width
                { get { return this.widthField; } set { this.widthField = value; } }

                [System.Xml.Serialization.XmlIgnoreAttribute()]
                public bool WidthSpecified
                { get { return this.widthFieldSpecified; } set { this.widthFieldSpecified = value; } }

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public int Height
                { get { return this.heightField; } set { this.heightField = value; } }

                [System.Xml.Serialization.XmlIgnoreAttribute()]
                public bool HeightSpecified
                { get { return this.heightFieldSpecified; } set { this.heightFieldSpecified = value; } }

                [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
                [System.SerializableAttribute()]
                [System.Diagnostics.DebuggerStepThroughAttribute()]
                [System.ComponentModel.DesignerCategoryAttribute("code")]
                [System.Xml.Serialization.XmlRootAttribute("Member", Namespace = "http://www.siemens.com/automation/Openness/SW/Interface/v2", IsNullable = false)]
                public partial class Member_T
                {
                    private object[] attributeListField;
                    private object[] itemsField;
                    private string nameField;
                    private string datatypeField;
                    private string versionField;
                    private Remanence_TE remanenceField;
                    private Accessibility_TE accessibilityField;
                    private bool informativeField;

                    public Member_T()
                    {
                        this.remanenceField = Remanence_TE.NonRetain;
                        this.accessibilityField = Accessibility_TE.Public;
                        this.informativeField = false;
                    }

                    [System.Xml.Serialization.XmlArrayItemAttribute("BooleanAttribute", typeof(BooleanAttribute_T), IsNullable = false)]
                    [System.Xml.Serialization.XmlArrayItemAttribute("IntegerAttribute", typeof(IntegerAttribute_T), IsNullable = false)]
                    [System.Xml.Serialization.XmlArrayItemAttribute("RealAttribute", typeof(RealAttribute_T), IsNullable = false)]
                    [System.Xml.Serialization.XmlArrayItemAttribute("StringAttribute", typeof(StringAttribute_T), IsNullable = false)]
                    public object[] AttributeList
                    { get { return this.attributeListField; } set { this.attributeListField = value; } }

                    [System.Xml.Serialization.XmlElementAttribute("Comment", typeof(Comment))]
                    [System.Xml.Serialization.XmlElementAttribute("Member", typeof(Member_T))]
                    [System.Xml.Serialization.XmlElementAttribute("Sections", typeof(Sections_T))]
                    [System.Xml.Serialization.XmlElementAttribute("StartValue", typeof(StartValue_T))]
                    [System.Xml.Serialization.XmlElementAttribute("Subelement", typeof(Subelement_T))]
                    public object[] Items
                    { get { return this.itemsField; } set { this.itemsField = value; } }

                    [System.Xml.Serialization.XmlAttributeAttribute()]
                    public string Name
                    { get { return this.nameField; } set { this.nameField = value; } }

                    [System.Xml.Serialization.XmlAttributeAttribute()]
                    public string Datatype
                    { get { return this.datatypeField; } set { this.datatypeField = value; } }
                }

                [System.Xml.Serialization.XmlAttributeAttribute()]

```

```

        public string Version
    {get{return this.versionField;}}set{this.versionField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    [System.ComponentModel.DefaultValueAttribute(Remanence_
TE.NonRetain)]
    public Remanence_TE Remanence
    {get{return this.remanenceField;}}set{this.remanenceField = valu
e;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    [System.ComponentModel.DefaultValueAttribute(Accessibil
ity_TE.Public)]
    public Accessibility_TE Accessibility
    {get{return this.accessibilityField;}}set{this.accessibilityFiel
d = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    [System.ComponentModel.DefaultValueAttribute(false)]
    public bool Informative
    {get{return this.informativeField;}}set{this.informativeField=va
lue;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Sections", Name
space = "http://www.siemens.com/automation/Openness/SW/Interfac
e/v2", IsNullable = false)]
    public partial class Sections_T : Node
    {
        private object[] attributeListField;
        private Section_T[] sectionField;
        private string datatypeField;
        private string versionField;

        [System.Xml.Serialization.XmlArrayItemAttribute("Boolea
nAttribute", typeof(BooleanAttribute_T), IsNullable = false)]
        [System.Xml.Serialization.XmlArrayItemAttribute("Intege
rAttribute", typeof(IntegerAttribute_T), IsNullable = false)]
        [System.Xml.Serialization.XmlArrayItemAttribute("RealAt
tribute", typeof(RealAttribute_T), IsNullable = false)]
        [System.Xml.Serialization.XmlArrayItemAttribute("String
Attribute", typeof(StringAttribute_T), IsNullable = false)]
        public object[] AttributeList
        {get{return this.attributeListField;}}set{this.attribute
ListField = value; }

        [System.Xml.Serialization.XmlElementAttribute("Section"
)]public Section_T[] Section
        {get{return this.sectionField;}}set{this.sectionField=va
lue;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Datatype
        {get{return this.datatypeField;}}set{this.datatypeField = value;
}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Version
        {get{return this.versionField;}}set{this.versionField = value;
}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Section", Names
pace = "http://www.siemens.com/automation/Openness/SW/Interface
/v2", IsNullable = false)]
    public partial class Section_T : Node
    {
        private Sections_T sectionsField;
        private Member_T[] memberField;
        private SectionName_TE nameField;

        public Sections_T Sections
        {get{return this.sectionsField;}}set{this.sectionsField=value}}

        [System.Xml.Serialization.XmlElementAttribute("Member")
]
        public Member_T[] Member
        {get{return this.memberField;}}set{this.memberField = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public SectionName_TE Name
        {get{return this.nameField;}}set{this.nameField = value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    public enum SectionName_TE
    {
        None,
        Input,
        Return,
        Output,
        InOut,
        Static,

```

```

        Temp,
        Constant,
        Base,}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("StartValue", Na
mespace = "", IsNullable = false)]
    public partial class StartValue_T
    {
        private string constantNameField;
        private bool isBulkValueField;
        private bool informativeField;
        private string valueField;

        public StartValue_T()
        {
            this.isBulkValueField = false;
            this.informativeField = false;
        }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string ConstantName
        {get{return this.constantNameField;}}set{this.constantNameField
= value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool IsBulkValue
        {get{return this.isBulkValueField;}}set{this.isBulkValueField =
value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool Informative
        {get{return this.informativeField;}}set{this.informativeField =
value;}}

        [System.Xml.Serialization.XmlTextAttribute()]
        public string Value
        {get{return this.valueField;}}set{this.valueField = value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Subelement", Na
mespace = "", IsNullable = false)]
    public partial class Subelement_T
    {
        private object[] itemsField;
        private string pathField;

        [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment))]
        [System.Xml.Serialization.XmlElementAttribute("StartVal
ue", typeof(StartValue_T))]
        public object[] Items
        {get{return this.itemsField;}}set{this.itemsField = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Path
        {get{return this.pathField;}}set{this.pathField = value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    public enum Remanence_TE
    {
        SetInIDB,
        NonRetain,
        Retain,}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    public enum Accessibility_TE
    {
        Public,
        Internal,
        Protected,
        Private,}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Access", Namesp
ace = "http://www.siemens.com/automation/Openness/SW/NetworkSou
rce/FlgNet/v1", IsNullable = false)]
    public partial class Access_T
    {
        private IntegerAttribute_T integerAttributeField;
        private object itemField;
        private object[] itemsField;
        private int uIdField;
        private Scope_TE scopeField;
        private bool uIdFieldSpecified;

```

```

        public IntegerAttribute_T IntegerAttribute
    {get{return this.integerAttributeField;}set{this.integerAttributeField=
value;}}

    [System.Xml.Serialization.XmlElementAttribute("Address"
, typeof(Address_T))]
    [System.Xml.Serialization.XmlElementAttribute("CallInfo"
, typeof(CallInfo_T))]
    [System.Xml.Serialization.XmlElementAttribute("Constant"
, typeof(Constant_T))]
    [System.Xml.Serialization.XmlElementAttribute("Expression"
, typeof(Expression_T))]
    [System.Xml.Serialization.XmlElementAttribute("Indirect"
, typeof(Indirect_T))]
    [System.Xml.Serialization.XmlElementAttribute("Instruction"
, typeof(Instruction_T))]
    [System.Xml.Serialization.XmlElementAttribute("Label"
, typeof(Label_T))]
    [System.Xml.Serialization.XmlElementAttribute("Statusword"
, typeof(Statusword_T))]
    [System.Xml.Serialization.XmlElementAttribute("Symbol"
, typeof(Symbol_T))]
    public object Item
    {get{return this.itemField;}set{this.itemField = value;}}

    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment_T))]
    [System.Xml.Serialization.XmlElementAttribute("LineComment"
, typeof(LineComment_T))]
    public object[] Items
    {get{return this.itemsField;}set{this.itemsField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public int Uid
    {get{return this.uidField;}set{this.uidField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public Scope_TE Scope
    {get{return this.scopeField;}set{this.scopeField = value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool UidSpecified
    {get{return this.uidFieldSpecified;}set{this.uidFieldSpecified
= value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Address", Names
pace = "http://www.siemens.com/automation/Openness/SW/NetworkSo
urce/FlgNet/v1", IsNullable = false)]
    public partial class Address_T
    {
        private Area_TE areaField;
        private string typeField;
        private int blockNumberField;
        private bool blockNumberFieldSpecified;
        private int bitOffsetField;
        private bool bitOffsetFieldSpecified;
        private bool informativeField;

        public Address_T()
        {this.informativeField = false; }

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public Area_TE Area
        {get{return this.areaField;}set{this.areaField = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Type
        {get{return this.typeField;}set{this.typeField = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public int BlockNumber
        {get{return this.blockNumberField;}set{this.blockNumberField =
value;}}

        [System.Xml.Serialization.XmlIgnoreAttribute()]
        public bool BlockNumberSpecified
        {get{return this.blockNumberFieldSpecified;}set{this.blockNumbe
rFieldSpecified = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public int BitOffset
        {get{return this.bitOffsetField;}set{this.bitOffsetField = valu
e; } }

        [System.Xml.Serialization.XmlIgnoreAttribute()]
        public bool BitOffsetSpecified
        {get{return this.bitOffsetFieldSpecified;}set{this.bitOffsetFieldSpecified =
value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        [System.ComponentModel.DefaultValueAttribute(false)]
        public bool Informative
        {get{return this.informativeField;}set{this.informativeField =
value;}}
    }

```

```

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    public enum Area_TE
    {
        None,
        Periphery,
        PeripheryInput,
        PeripheryOutput,
        Input,
        Output,
        Memory,
        FB,
        FC,
        DB,
        DI,
        Timer,
        Counter,
        Local,}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlTypeAttribute(AnonymousType =
true)]
    [System.Xml.Serialization.XmlRootAttribute(Namespace = "htt
p://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet
/v1", IsNullable = false)]
    public partial class CallInfo : CallInfo_T {}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    public partial class CallInfo_T
    {
        private IntegerAttribute_T integerAttributeField;
        private DateAttribute_T dateAttributeField;
        private object[] itemsField;
        private Token_T tokenField;
        private object[] items1Field;
        private Instance_T instanceField;
        private Parameter_T[] parameterField;
        private string nameField;
        private BlockType_TE blockTypeField;

        [System.Xml.Serialization.XmlElementAttribute(Order = 0
)]
        public IntegerAttribute_T IntegerAttribute
        {get{return this.integerAttributeField;}set{this.integerAttribu
teField = value;}}

        [System.Xml.Serialization.XmlElementAttribute(Order = 1
)]
        public DateAttribute_T DateAttribute
        {get{return this.dateAttributeField;}set{this.dateAttributeFiel
d = value;}}

        [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment), Order = 2)]
        [System.Xml.Serialization.XmlElementAttribute("LineComm
ent", typeof(LineComment_T), Order = 2)]
        public object[] Items
        {get{return this.itemsField;}set{this.itemsField = value;}}

        [System.Xml.Serialization.XmlElementAttribute(Order = 3
)]
        public Token_T Token
        {get{return this.tokenField;}set{this.tokenField = value;}}

        [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment), Order = 4)]
        [System.Xml.Serialization.XmlElementAttribute("LineComm
ent", typeof(LineComment_T), Order = 4)]
        public object[] Items1
        {get{return this.items1Field;}set{this.items1Field = value;}}

        [System.Xml.Serialization.XmlElementAttribute(Order = 5
)]
        public Instance_T Instance
        {get{return this.instanceField;}set{this.instanceField = value;
}}

        [System.Xml.Serialization.XmlElementAttribute("Paramete
r", Order = 6)]
        public Parameter_T[] Parameter
        {get{return this.parameterField;}set{this.parameterField = valu
e;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Name
        {get{return this.nameField;}set{this.nameField = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public BlockType_TE BlockType
        {get{return this.blockTypeField;}set{this.blockTypeField = valu
e;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]

```

```

[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Instance", Name
space = "http://www.siemens.com/automation/Openness/SW/NetworkS
ource/FlgNet/v1", IsNullable = false)]
public partial class Instance_T
{
    private object[] itemsField;
    private object[] items1Field;
    private Address_T addressField;
    private int uIdField;
    private bool uIdFieldSpecified;
    private Scope_TE scopeField;

    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute("Absolute
Offset", typeof(AbsoluteOffset_T))]
    [System.Xml.Serialization.XmlElementAttribute("Componen
t", typeof(Component_T))]
    [System.Xml.Serialization.XmlElementAttribute("Token",
typeof(Token_T))]
    public object[] Items
    {get{return this.itemsField;}set{this.itemsField = value;}}

    [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment))]
    [System.Xml.Serialization.XmlElementAttribute("LineComm
ent", typeof(LineComment_T))]
    public object[] Items1
    {get{return this.items1Field;}set{this.items1Field = value;}}

    public Address_T Address
    {get{return this.addressField;}set{this.addressField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public int UIId
    {get{return this.uIdField;}set{this.uIdField = value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool UIIdSpecified
    {get{return this.uIdFieldSpecified;}set{this.uIdFieldSpecified
= value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public Scope_TE Scope
    {get{return this.scopeField;}set{this.scopeField = value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("AbsoluteOffset"
, Namespace = "", IsNullable = false)]
    public partial class AbsoluteOffset_T
    {
        private int bitOffsetField;

        private string typeField;

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public int BitOffset
    {get{return this.bitOffsetField;}set{this.bitOffsetField = valu
e;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Type
    {get{return this.typeField;}set{this.typeField = value;}}

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("Component", Nam
espace = "http://www.siemens.com/automation/Openness/SW/Network
Source/FlgNet/v1", IsNullable = false)]
        public partial class Component_T
        {
            private Access_T[] accessField;
            private string nameField;
            private string sliceAccessModifierField;
            private string simpleAccessModifierField;

            public Component_T()
            {
                this.sliceAccessModifierField = "undef";
                this.simpleAccessModifierField = "None";
            }

            [System.Xml.Serialization.XmlElementAttribute("Access")]
            public Access_T[] Access
            {get{return this.accessField;}set{this.accessField = value;}}

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public string Name
            {get{return this.nameField;}set{this.nameField = value;}}

            [System.Xml.Serialization.XmlAttributeAttribute()]
            [System.ComponentModel.DefaultValueAttribute("undef")]
            public string SliceAccessModifier

[get{return this.sliceAccessModifierField;}set{this.sliceAccess
ModifierField = value;}}

            [System.Xml.Serialization.XmlAttributeAttribute()]
            [System.ComponentModel.DefaultValueAttribute("None")]
            public string SimpleAccessModifier
            {get{return this.simpleAccessModifierField;}set{this.simpleAcce
ssModifierField = value;}}

            [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
            [System.SerializableAttribute()]
            [System.Xml.Serialization.XmlRootAttribute("Parameter", Nam
espace = "", IsNullable = false)]
            public partial class Parameter_T
            {
                private IntegerAttribute_T integerAttributeField;
                private StringAttribute_T stringAttributeField;
                private object[] itemsField;
                private Access_T accessField;
                private object[] items1Field;
                private string nameField;
                private SectionName_TE sectionField;
                private bool sectionFieldSpecified;
                private string typeField;
                private string templateReferenceField;

                [System.Xml.Serialization.XmlElementAttribute(Order = 0
)]
                public IntegerAttribute_T IntegerAttribute
                {get{return this.integerAttributeField;}set{this.integerAttribu
teField = value;}}

                [System.Xml.Serialization.XmlElementAttribute(Order = 1
)]
                public StringAttribute_T StringAttribute
                {get{return this.stringAttributeField;}set{this.stringAttribute
Field = value;}}

                [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment), Order = 2)]
                [System.Xml.Serialization.XmlElementAttribute("LineComm
ent", typeof(LineComment_T), Order = 2)]
                public object[] Items
                {get{return this.itemsField;}set{this.itemsField = value;}}

                [System.Xml.Serialization.XmlElementAttribute(Order = 3
)]
                public Access_T Access
                {get{return this.accessField;}set{this.accessField = value;}}

                [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment), Order = 4)]
                [System.Xml.Serialization.XmlElementAttribute("LineComm
ent", typeof(LineComment_T), Order = 4)]
                public object[] Items1
                {get{return this.items1Field;}set{this.items1Field = value;}}

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public string Name
                {get{return this.nameField;}set{this.nameField = value;}}

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public SectionName_TE Section
                {get{return this.sectionField;}set{this.sectionField = value;}}

                /// <remarks/>
                [System.Xml.Serialization.XmlIgnoreAttribute()]
                public bool SectionSpecified
                {get{return this.sectionFieldSpecified;}set{this.sectionFieldSp
ecified = value;}}

                /// <remarks/>
                [System.Xml.Serialization.XmlAttributeAttribute()]
                public string Type
                {get{return this.typeField;}set{this.typeField = value;}}

```

```

[System.Xml.Serialization.XmlAttributeAttribute()]
public string TemplateReference
{get{return this.templateReferenceField;}}set{this.templateRefer
enceField = value;}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
public enum BlockType_TE
{
    DB,
    FB,
    FC,
    OB,
    UDT,
    FBT,
    FCT,}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Constant", Name
space = "", IsNullable = false)]
public partial class Constant_T
{
    private ConstantType constantTypeField;
    private ConstantValue constantValueField;
    private StringAttribute_T[] stringAttributeField;
    private string nameField;

    public ConstantType ConstantType
    {get{return this.constantTypeField;}}set{this.constantTypeField
= value;}}

    public ConstantValue ConstantValue
    {get{return this.constantValueField;}}set{this.constantValueField = value;}}

    [System.Xml.Serialization.XmlElementAttribute("StringAt
tribute")]
    public StringAttribute_T[] StringAttribute
    {get{return this.stringAttributeField;}}set{this.stringAttribute
Field = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Name
    {get{return this.nameField;}}set{this.nameField = value;}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType =
true)]
[System.Xml.Serialization.XmlRootAttribute(Namespace = "",
IsNullable = false)]
public partial class ConstantType
{
    private bool informativeField;
    private bool informativeFieldSpecified;
    private string valueField;

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public bool Informative
    {get{return this.informativeField;}}set{this.informativeField =
value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool InformativeSpecified
    {get{return this.informativeFieldSpecified;}}set{this.informativ
eFieldSpecified = value;}}

    [System.Xml.Serialization.XmlTextAttribute()]
    public string Value
    {get{return this.valueField;}}set{this.valueField = value;}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlTypeAttribute(AnonymousType =
true)]
[System.Xml.Serialization.XmlRootAttribute(Namespace = "",
IsNullable = false)]
public partial class ConstantValue : ConstantValue_T
{
    private bool informativeField;
    private bool informativeFieldSpecified;

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public bool Informative
    {get{return this.informativeField;}}set{this.informativeField =
value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool InformativeSpecified
    {get{return this.informativeFieldSpecified;}}set{this.informativ
eFieldSpecified = value;}}

```

```

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
public partial class ConstantValue_T
{
    private string valueField;

    [System.Xml.Serialization.XmlTextAttribute()]
    public string Value
    {get{return this.valueField;}}set{this.valueField = value;}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Expression", Na
mespace = "", IsNullable = false)]
public partial class Expression_T
{
    private object[] itemsField;
    private object[] items1Field;
    private int uIdField;
    private bool uIdFieldSpecified;

    [System.Xml.Serialization.XmlElementAttribute("Access",
typeof(Access_T))]
    [System.Xml.Serialization.XmlElementAttribute("Token",
typeof(Token_T))]
    public object[] Items
    {get{return this.itemsField;}}set{this.itemsField = value;}}

    [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment))]
    [System.Xml.Serialization.XmlElementAttribute("LineComm
ent", typeof(LineComment_T))]
    public object[] Items1
    {get{return this.items1Field;}}set{this.items1Field = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public int Uid
    {get{return this.uIdField;}}set{this.uIdField = value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool UidSpecified
    {get{return this.uIdFieldSpecified;}}set{this.uIdFieldSpecified
= value;}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Indirect", Name
space = "", IsNullable = false)]
public partial class Indirect_T
{
    private object[] itemsField;
    private Access_T accessField;
    private Width_TE widthField;
    private Area_TE areaField;
    private bool areaFieldSpecified;

    private Register_TE registerField;
    private bool registerFieldSpecified;
    private string bitOffsetField;

    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute("Comment"
, typeof(Comment))]
    [System.Xml.Serialization.XmlElementAttribute("LineComm
ent", typeof(LineComment_T))]
    [System.Xml.Serialization.XmlElementAttribute("Token",
typeof(Token_T))]
    public object[] Items
    {get{return this.itemsField;}}set{this.itemsField = value;}}

    /// <remarks/>
    public Access_T Access
    {get{return this.accessField;}}set{this.accessField = value;}}

    /// <remarks/>
    [System.Xml.Serialization.XmlAttributeAttribute()]
    public Width_TE Width
    {get{return this.widthField;}}set{this.widthField = value;}}

    /// <remarks/>
    [System.Xml.Serialization.XmlAttributeAttribute()]
    public Area_TE Area
    {get{return this.areaField;}}set{this.areaField = value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool AreaSpecified
    {get{return this.areaFieldSpecified;}}set{this.areaFieldSpecifie
d = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public Register_TE Register
    {get{return this.registerField;}}set{this.registerField = value;
}}

```

```

[System.Xml.Serialization.XmlIgnoreAttribute()]
public bool RegisterSpecified
{get{return this.registerFieldSpecified;}set{this.registerFieldSpecified = value;}}

[System.Xml.Serialization.XmlAttributeAttribute()]
public string BitOffset
{get{return this.bitOffsetField;}set{this.bitOffsetField = value;}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
public enum Width_TE
{
    None,
    Bit,
    Byte,
    Word,
    Offset,
    Double,
    Pointer,
    Long,
    Any,
    Block,}

/// <remarks/>
[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
public enum Register_TE
{
    AR1,
    AR2,}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Instruction", Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1", IsNullable = false)]
public partial class Instruction_T
{
    private object[] itemsField;
    private Token_T tokenField;
    private object[] items1Field;
    private TemplateValue_T[] templateValueField;
    private Instance_T instanceField;
    private Parameter_T[] parameterField;
    private string nameField;
    private string versionField;

    [System.Xml.Serialization.XmlElementAttribute("Comment", typeof(Comment), Order = 0)]
    [System.Xml.Serialization.XmlElementAttribute("LineComment", typeof(LineComment_T), Order = 0)]
    public object[] Items
    {get{return this.itemsField;}set{this.itemsField=value;}}

    [System.Xml.Serialization.XmlElementAttribute(Order = 1)]
    public Token_T Token
    {get{return this.tokenField;}set{this.tokenField = value;}}

    [System.Xml.Serialization.XmlElementAttribute("Comment", typeof(Comment), Order = 2)]
    [System.Xml.Serialization.XmlElementAttribute("LineComment", typeof(LineComment_T), Order = 2)]
    public object[] Items1
    {get{return this.items1Field;}set{this.items1Field=value;}}

    [System.Xml.Serialization.XmlElementAttribute("TemplateValue", Order = 3)]
    public TemplateValue_T[] TemplateValue
    {get{return this.templateValueField;}set{this.templateValueField = value;}}

    [System.Xml.Serialization.XmlElementAttribute(Order = 4)]
    public Instance_T Instance
    {get{return this.instanceField;}set{this.instanceField = value;}}

    [System.Xml.Serialization.XmlElementAttribute("Parameter", Order = 5)]
    public Parameter_T[] Parameter
    {get{return this.parameterField;}set{this.parameterField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Name
    {get{return this.nameField;}set{this.nameField = value;}}

    /// <remarks/>
    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Version
    {get{return this.versionField;}set{this.versionField = value;}}
}

```

```

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("TemplateValue", Namespace = "", IsNullable = false)]
public partial class TemplateValue_T
{
    private string nameField;

    private TemplateType_TE typeField;

    private string valueField;

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Name
    {get{return this.nameField;}set{this.nameField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public TemplateType_TE Type
    {get{return this.typeField;}set{this.typeField=value;}}

    /// <remarks/>
    [System.Xml.Serialization.XmlTextAttribute()]
    public string Value
    {get{return this.valueField;}set{this.valueField=value;}}

    /// <remarks/>
    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    public enum TemplateType_TE
    {
        /// <remarks/>
        Cardinality,
        /// <remarks/>
        Type,
        /// <remarks/>
        Operation,}

    /// <remarks/>
    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Label", Namespace = "", IsNullable = false)]
    public partial class Label_T
    {
        private string nameField;

        /// <remarks/>
        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Name
        {get{return this.nameField;}set{this.nameField = value;}}

        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("Statusword", Namespace = "", IsNullable = false)]
        public partial class Statusword_T
        {
            private Statusword_TE combinationField;

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public Statusword_TE Combination
            {get{return this.combinationField;}set{this.combinationField = value;}}

            [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
            [System.SerializableAttribute()]
            public enum Statusword_TE
            {
                BR,
                OV,
                OS,
                EQ,
                NE,
                GT,
                LT,
                GE,
                LE,
                UO,
                NU,
                STW,}

            [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
            [System.SerializableAttribute()]
            [System.Diagnostics.DebuggerStepThroughAttribute()]
            [System.ComponentModel.DesignerCategoryAttribute("code")]

```

```

[System.Xml.Serialization.XmlRootAttribute("Symbol", Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1", IsNullable = false)]
public partial class Symbol_T
{
    private object[] itemsField;
    private object[] items1Field;
    private Address_T addressField;
    private int uIdField;
    private bool uIdFieldSpecified;
    private Scope_TE scopeField;
    private bool scopeFieldSpecified;

    [System.Xml.Serialization.XmlElementAttribute("AbsoluteOffset", typeof(AbsoluteOffset_T))]
    [System.Xml.Serialization.XmlElementAttribute("Component", typeof(Component_T))]
    [System.Xml.Serialization.XmlElementAttribute("Token", typeof(Token_T))]
    public object[] Items
    {get{return this.itemsField;}set{this.itemsField = value;}}

    [System.Xml.Serialization.XmlElementAttribute("Comment", typeof(Comment))]
    [System.Xml.Serialization.XmlElementAttribute("LineComment", typeof(LineComment_T))]
    public object[] Items1
    {get{return this.items1Field;}set{this.items1Field = value;}}

    public Address_T Address
    {get{return this.addressField;}set{this.addressField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public int UId
    {get{return this.uIdField;}set{this.uIdField = value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool UIdSpecified
    {get{return this.uIdFieldSpecified;}set{this.uIdFieldSpecified = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public Scope_TE Scope
    {get{return this.scopeField;}set{this.scopeField = value;}}

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool ScopeSpecified
    {get{return this.scopeFieldSpecified;}set{this.scopeFieldSpecified = value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("LabelDeclaration", Namespace = "", IsNullable = false)]
    public partial class LabelDeclaration_T
    {
        private IntegerAttribute_T integerAttributeField;
        private Label_T labelField;
        private object[] itemsField;
        private Token_T tokenField;
        private object[] items1Field;
        private int uIdField;
        private bool uIdFieldSpecified;

        [System.Xml.Serialization.XmlElementAttribute(Order = 0)]
        public IntegerAttribute_T IntegerAttribute
        {get{return this.integerAttributeField;}set{this.integerAttributeField = value;}}

        [System.Xml.Serialization.XmlElementAttribute(Order = 1)]
        public Label_T Label
        {get{return this.labelField;}set{this.labelField = value;}}

        [System.Xml.Serialization.XmlElementAttribute("Comment", typeof(Comment), Order = 2)]
        [System.Xml.Serialization.XmlElementAttribute("LineComment", typeof(LineComment_T), Order = 2)]
        public object[] Items
        {get{return this.itemsField;}set{this.itemsField = value;}}

        [System.Xml.Serialization.XmlElementAttribute(Order = 3)]
        public Token_T Token
        {get{return this.tokenField;}set{this.tokenField = value;}}

        [System.Xml.Serialization.XmlElementAttribute("Comment", typeof(Comment), Order = 4)]
        [System.Xml.Serialization.XmlElementAttribute("LineComment", typeof(LineComment_T), Order = 4)]
        public object[] Items1
        {get{return this.items1Field;}set{this.items1Field = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public int UId
        {get{return this.uIdField;}set{this.uIdField = value;}}
    }

    [System.Xml.Serialization.XmlIgnoreAttribute()]
    public bool UIdSpecified
    {get{return this.uIdFieldSpecified;}set{this.uIdFieldSpecified = value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Powerrail", Namespace = "", IsNullable = false)]
    public partial class Powerrail_T {}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Openbranch", Namespace = "", IsNullable = false)]
    public partial class Openbranch_T {}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("OpenCon", Namespace = "", IsNullable = false)]
    public partial class OpenCon_T
    {
        private int uIdField;

        /// <remarks/>
        [System.Xml.Serialization.XmlAttributeAttribute()]
        public int UId
        {get{return this.uIdField;}set{this.uIdField = value;}}

        /// <remarks/>
        [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
        [System.SerializableAttribute()]
        [System.Diagnostics.DebuggerStepThroughAttribute()]
        [System.ComponentModel.DesignerCategoryAttribute("code")]
        [System.Xml.Serialization.XmlRootAttribute("Call", Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1", IsNullable = false)]
        public partial class Call_T
        {
            private CallInfo callInfoField;
            private TemplateValue_T[] templateValueField;
            private AutomaticTyped_T[] automaticTypedField;
            private Invisible_T[] invisibleField;
            private Neg_T[] negatedField;
            private Comment commentField;
            private int uIdField;

            /// <remarks/>
            public CallInfo CallInfo
            {get{return this.callInfoField;}set{this.callInfoField = value;}}

            [System.Xml.Serialization.XmlElementAttribute("TemplateValue")]
            public TemplateValue_T[] TemplateValue
            {get{return this.templateValueField;}set{this.templateValueField = value;}}

            [System.Xml.Serialization.XmlElementAttribute("AutomaticTyped")]
            public AutomaticTyped_T[] AutomaticTyped
            {get{return this.automaticTypedField;}set{this.automaticTypedField = value;}}

            [System.Xml.Serialization.XmlElementAttribute("Invisible")]
            public Invisible_T[] Invisible
            {get{return this.invisibleField;}set{this.invisibleField = value;}}

            [System.Xml.Serialization.XmlElementAttribute("Negated")]
            public Neg_T[] Negated
            {get{return this.negatedField;}set{this.negatedField = value;}}

            public Comment Comment
            {get{return this.commentField;}set{this.commentField = value;}}

            [System.Xml.Serialization.XmlAttributeAttribute()]
            public int UId
            {get{return this.uIdField;}set{this.uIdField = value;}}

            [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6.1055.0")]
            [System.SerializableAttribute()]
            [System.Diagnostics.DebuggerStepThroughAttribute()]
            [System.ComponentModel.DesignerCategoryAttribute("code")]
            [System.Xml.Serialization.XmlRootAttribute("AutomaticTyped", Namespace = "", IsNullable = false)]
            public partial class AutomaticTyped_T
            {
                private string nameField;

                [System.Xml.Serialization.XmlAttributeAttribute()]
                public string Name
                {get{return this.nameField;}}
            }

```

```

        set{this.nameField=value;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Invisible", Name
space = "", IsNullable = false)]
public partial class Invisible_T
{
    private string nameField;

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Name{get{return this.nameField;}set{this.
nameField=value;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Negated", Names
pace = "", IsNullable = false)]
public partial class Neg_T
{
    private string nameField;

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Name
    {get{return this.nameField;}
}set{this.nameField = value;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("FlgNet", Namesp
ace = "http://www.siemens.com/automation/Openness/SW/NetworkSou
rce/FlgNet/v1", IsNullable = false)]
public partial class FlgNet_T : Node
{
    private Labels_T labelsField;
    private object[] partsField;
    private Wires_T wiresField;
    public Labels_T Labels
    {get{return this.labelsField;}
set{this.labelsField = value;}}

    [System.Xml.Serialization.XmlArrayItemAttribute("Access
", typeof(Access_T), IsNullable = false)]
    [System.Xml.Serialization.XmlArrayItemAttribute("Call",
typeof(Call_T), IsNullable = false)]
    [System.Xml.Serialization.XmlArrayItemAttribute("Part",
typeof(Part_T), IsNullable = false)]
    public object[] Parts{get{return this.partsField;}
set{this.partsField = value;}}

    public Wires_T Wires
    {get{return this.wiresField;}set{this.wiresField = valu
e;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Labels", Namesp
ace = "", IsNullable = false)]
public partial class Labels_T
{
    private LabelDeclaration_T[] labelDeclarationField;

    [System.Xml.Serialization.XmlElementAttribute("LabelDec
laration")]
    public LabelDeclaration_T[] LabelDeclaration
    {get{return this.labelDeclarationField;}set{this.labelD
eclarationField = value;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Part", Namespac
e = "http://www.siemens.com/automation/Openness/SW/NetworkSour
ce/FlgNet/v1", IsNullable = false)]
public partial class Part_T
{
    private object itemField;
    private TemplateValue_T[] templateValueField;
    private AutomaticTyped_T[] automaticTypedField;
    private Invisible_T[] invisibleField;
    private Neg_T[] negatedField;
    private Comment commentField;
    private int uidField;
    private string nameField;
    private string versionField;
    private bool disabledENOFfield;

    public Part_T()

```

```

    {
        this.disabledENOFfield = false;}

    [System.Xml.Serialization.XmlElementAttribute("Equation
", typeof(Equation_T))]
    [System.Xml.Serialization.XmlElementAttribute("Instance
", typeof(Instance_T))]
    public object Item
    {get{return this.itemField;}set{this.itemField=value}}

    [System.Xml.Serialization.XmlElementAttribute("Template
Value")]
    public TemplateValue_T[] TemplateValue{get{return this.
templateValueField; }set{this.templateValueField = value; }}

    [System.Xml.Serialization.XmlElementAttribute("Automati
cTyped")]
    public AutomaticTyped_T[] AutomaticTyped
    {get{return this.automaticTypedField;}set{this.automaticTypedFi
eld=value;}}

    [System.Xml.Serialization.XmlElementAttribute("Invisibl
e")]
    public Invisible_T[] Invisible{get{return this.invisibl
eField;}set{this.invisibleField = value;}}

    [System.Xml.Serialization.XmlElementAttribute("Negated"
)]
    public Neg_T[] Negated{get{return this.negatedField;}
set{this.negatedField = value;}}

    public Comment Comment{get{return this.commentField;}
set{this.commentField=value;}} }

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public int Uid
    {get{return this.uidField;}set{this.uidField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Name{get{return this.nameField;}set{
this.nameField = value;}}
    [System.Xml.Serialization.XmlAttributeAttribute()]
    public string Version{get{return this.versionField;}
set{this.versionField = value;}}

    [System.Xml.Serialization.XmlAttributeAttribute()]
    [System.ComponentModel.DefaultValueAttribute(false)]
    public bool DisabledENO{get{return this.disabledENOFiel
d;}set{this.disabledENOFfield = value;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Equation", Name
space = "", IsNullable = false)]
public partial class Equation_T
{
    private string valueField;

    [System.Xml.Serialization.XmlTextAttribute()]
    public string Value
    {get{return this.valueField; }
set{this.valueField = value;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Wires", Namespa
ce = "http://www.siemens.com/automation/Openness/SW/NetworkSour
ce/FlgNet/v1", IsNullable = false)]
public partial class Wires_T
{
    private Wire_T[] wireField;

    [System.Xml.Serialization.XmlElementAttribute("Wire")]
    public Wire_T[] Wire{get{return this.wireField;}set
{this.wireField=value;}}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("Wire", Namespac
e = "http://www.siemens.com/automation/Openness/SW/NetworkSour
ce/FlgNet/v1", IsNullable = false)]
public partial class Wire_T
{
    private object[] itemsField;
    private int uidField;

    [System.Xml.Serialization.XmlElementAttribute("IdentCon
", typeof(IdentCon_T))]
    [System.Xml.Serialization.XmlElementAttribute("NameCon"
, typeof(NameCon_T))]
    [System.Xml.Serialization.XmlElementAttribute("OpenCon"
, typeof(OpenCon_T))]

```

```

[System.Xml.Serialization.XmlElementAttribute("Openbranch", typeof(Openbranch_T))]
[System.Xml.Serialization.XmlElementAttribute("Powerrail", typeof(Powerrail_T))]
public object[] Items{get{return this.itemsField;}
set{this.itemsField = value;}}

[System.Xml.Serialization.XmlAttributeAttribute()]
public int UId
{get{return this.uIdField;}set{this.uIdField = value;}}

[System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
[System.SerializableAttribute()]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Xml.Serialization.XmlRootAttribute("IdentCon", Name
space = "", IsNullable = false)]
public partial class IdentCon_T
{
    private int uIdField;

    [System.Xml.Serialization.XmlAttributeAttribute()]
    public int UId
    {get{return this.uIdField;}set{this.uIdField=value;}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("NameCon", Name
pace = "", IsNullable = false)]
    public partial class NameCon_T
    {

        private int uIdField;
        private string nameField;

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public int UId{get{return this.uIdField;}set{this.uIdFi
eld = value;}}

        [System.Xml.Serialization.XmlAttributeAttribute()]
        public string Name
        {get{return this.nameField;}
set{this.nameField = value; }}}

    [System.CodeDom.Compiler.GeneratedCodeAttribute("xsd", "4.6
.1055.0")]
    [System.SerializableAttribute()]
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Xml.Serialization.XmlRootAttribute("Parts", Namespa
ce = "http://www.siemens.com/automation/Openness/SW/NetworkSour
ce/FlgNet/v1", IsNullable = false)]
    public partial class Parts_T
    {
        private object[] itemsField;

        [System.Xml.Serialization.XmlElementAttribute("Access",
typeof(Access_T))]
        [System.Xml.Serialization.XmlElementAttribute("Call", t
ypeof(Call_T))]
        [System.Xml.Serialization.XmlElementAttribute("Part", t
ypeof(Part_T))]
        public object[] Items{get{return this.itemsField;}
set{this.itemsField = value;}}

#endregion

```

Funtion Blocks

```
using System.Xml;
using System.Xml.Serialization;

namespace AppGeneratorFB
{
    #region ADDED CLASSES

    public abstract class Node
    {
        [XmlElement]
        public XmlElement[] OtherElements { get; set; }
        [XmlAnyAttribute]
        public XmlAttribute[] OtherAttributes { get; set; }
    }

    // Classes inherit from Node
    public class Document : Node
    {
        [XmlElement(ElementName = "SW.Blocks.FB")]
        public Blocks Blocks_ { get; set; }
    }

    public class Blocks : Node
    {
        [XmlElement("AttributeList", Namespace = "")]
        public AttributList_T AttributeList { get; set; }
        [XmlElement("ObjectList", Namespace = "")]
        public ObjectList_T ObjectList { get; set; }
        [XmlAttribute("ID")]
        public string ID { get; set; }
    }

    [XmlRoot(Namespace = "http://www.siemens.com/automation/Openness/SW/Interface/v2")]
    public class Interface : Node
    {
        public Sections_T Sections { get; set; }
    }

    [System.Xml.Serialization.XmlRootAttribute("ObjectList", Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1", IsNullable = false)]
    public class ObjectList_T : Node
    {
        [XmlElement("SW.Blocks.CompileUnit", Namespace = "")]
        public SWCompile_T[] SWCompile { get; set; }
    }

    [XmlRoot(Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1")]
    public class SWCompile_T : Node
    {
        [XmlElement("AttributeList", Namespace = "")]
        public AttributList_T AttributeList { get; set; }
        [XmlElement("ObjectList", Namespace = "")]
        public ObjectList_T ObjectList { get; set; }
        [XmlAttribute("ID")]
        public string ID { get; set; }
        [XmlAttribute("CompositionName")]
        public string CompName { get; set; }
    }

    public class NetSource_T
    {
        [XmlElement("FlgNet", Namespace = "http://www.siemens.com/automation/Openness/SW/NetworkSource/FlgNet/v1")]
        public FlgNet_T flgNet { get; set; }
    }
    #endregion

    #region AUTO-GENERATED CLASSES → THE SAME USED FOR THE Organization Block

```

Global Database Blocks

```
using System.Xml;
using System.Xml.Serialization;

namespace AppGeneratorGlobalDB
{
    #region CLASSES ADDED MANUALLY

    public abstract class Node
    {
        [XmlElement]
        public XmlElement[] OtherElements { get; set; }
        [XmlAnyAttribute]
        public XmlAttribute[] OtherAttributes { get; set; }
    }

    // Classes inherit from Node
    public class Document : Node
    {
        [XmlElement(ElementName = "SW.Blocks.GlobalDB")]
        public Blocks Blocks_ { get; set; }
    }

    public class Blocks : Node
    {
        [XmlElement("AttributeList", Namespace = "")]
        public AttributList_T AttributeList { get; set; }
        [XmlAttribute("ID")]
        public string ID { get; set; }
    }

    [XmlRoot(Namespace = "http://www.siemens.com/automation/Openness/SW/Interface/v2")]
    public class Interface : Node
    {
        public Sections_T Sections { get; set; }
    }

    public class Name_T
    {
        [XmlText]
        public string Nmbr { get; set; }
    }
    #endregion

    #region AUTO-GENERATED CLASSES → THE SAME USED FOR THE Organization Block

```

Instance Database Blocks

```
using System.Xml;
using System.Xml.Serialization;

namespace AppGeneratorDB
{
    #region ADDED CLASSES

    public abstract class Node
    {
        [XmlElement]
        public XmlElement[] OtherElements { get; set; }
        [XmlAnyAttribute]
        public XmlAttribute[] OtherAttributes { get; set; }
    }

    // Classes inherit from Node
    public class Document : Node
    {
        [XmlElement(ElementName = "SW.Blocks.InstanceDB")]
        public Blocks Blocks_ { get; set; }
    }

    public class Blocks : Node
    {
        [XmlElement("AttributeList", Namespace = "")]
        public AttributList_T AttributeList { get; set; }
        [XmlAttribute("ID")]
        public string ID { get; set; }
    }

    [XmlRoot(Namespace = "http://www.siemens.com/automation/Openness/SW/Interface/v2")]
    public class Interface : Node
    {
        public Sections_T Sections { get; set; }
    }

    public class Name_T
    {
        [XmlText]
        public string Nmbr { get; set; }
    }
    #endregion

    #region AUTO-GENERATED CLASSES → THE SAME USED FOR THE Organization Block

```

Tagtables

```

using System.Xml;
using System.Xml.Serialization;

namespace AppGeneratorTagTables
{
    #region ADDED CLASSES

    public abstract class Node
    {
        [XmlElement]
        public XmlElement[] OtherElements { get; set; }
        [XmlAttribute]
        public XmlAttribute[] OtherAttributes { get; set; }
    }

    // Classes inherit from Node
    public class Document : Node
    {
        [XmlElement(ElementName = "SW.Tags.PlcTagTable")]
        public Blocks Blocks_ { get; set; }
    }

    public class Blocks : Node
    {
        [XmlElement("AttributeList", Namespace = "")]
        public AttributeList_T AttributeList { get; set; }
        [XmlElement("ObjectList", Namespace = "")]
        public ObjectList_T ObjectList { get; set; }
        [XmlAttribute("ID")]
        public string ID { get; set; }
    }

    }

    [System.Xml.Serialization.XmlRootAttribute("ObjectList", Namespace = "", IsNullable = false)]
    public class ObjectList_T : Node
    {
        [XmlElement("SW.Tags.PlcTag", Namespace = "")]
        public SWTags_T[] SWTags { get; set; }
    }

    [XmlRoot(Namespace = "")]
    public class SWTags_T : Node
    {
        [XmlElement("AttributeList", Namespace = "")]
        public AttributeList_T AttributeList { get; set; }
        [XmlElement("ObjectList", Namespace = "")]
        public ObjectList_T ObjectList { get; set; }
        [XmlAttribute("ID")]
        public string ID { get; set; }
        [XmlAttribute("CompositionName")]
        public string CompName { get; set; }
    }

    #endregion

    #region AUTO-GENERATED CLASSES → THE SAME USED FOR THE
    Organization Block
    
```

ANEXO B: Programa desenvolvido no TIA Portal

Graficets dos *function blocks* desenvolvidos

Conv_i

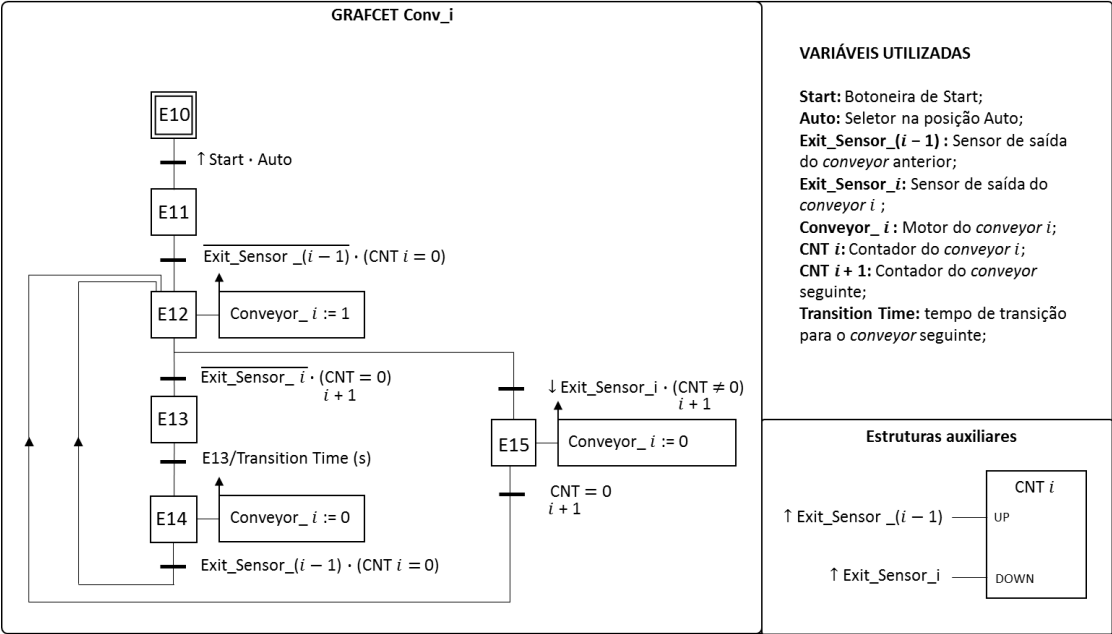


Figura 1 – GRAFCET do *function block* Conv_i.

Conv_n

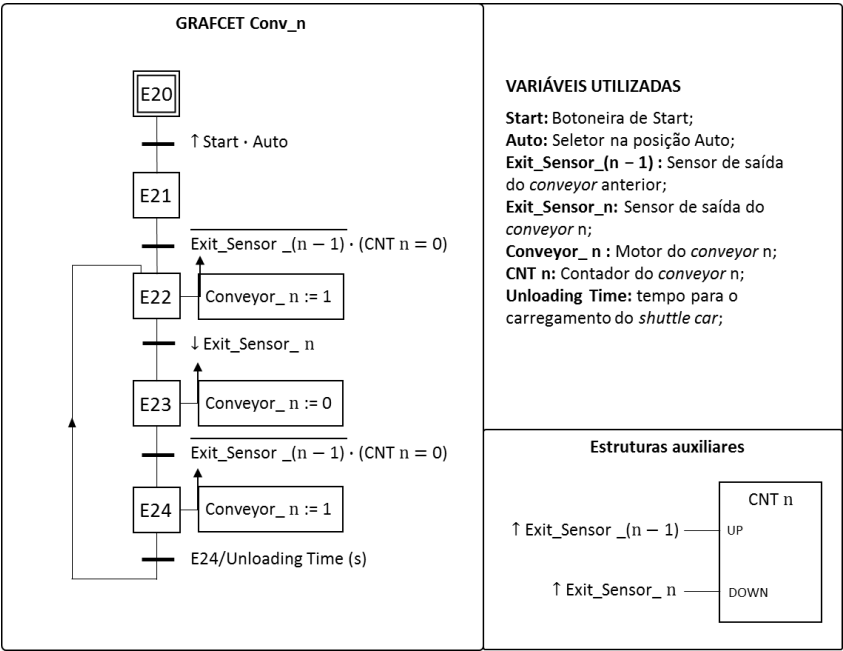


Figura 2 – GRAFCET do *function block* Conv_n.

Shuttle_Manual

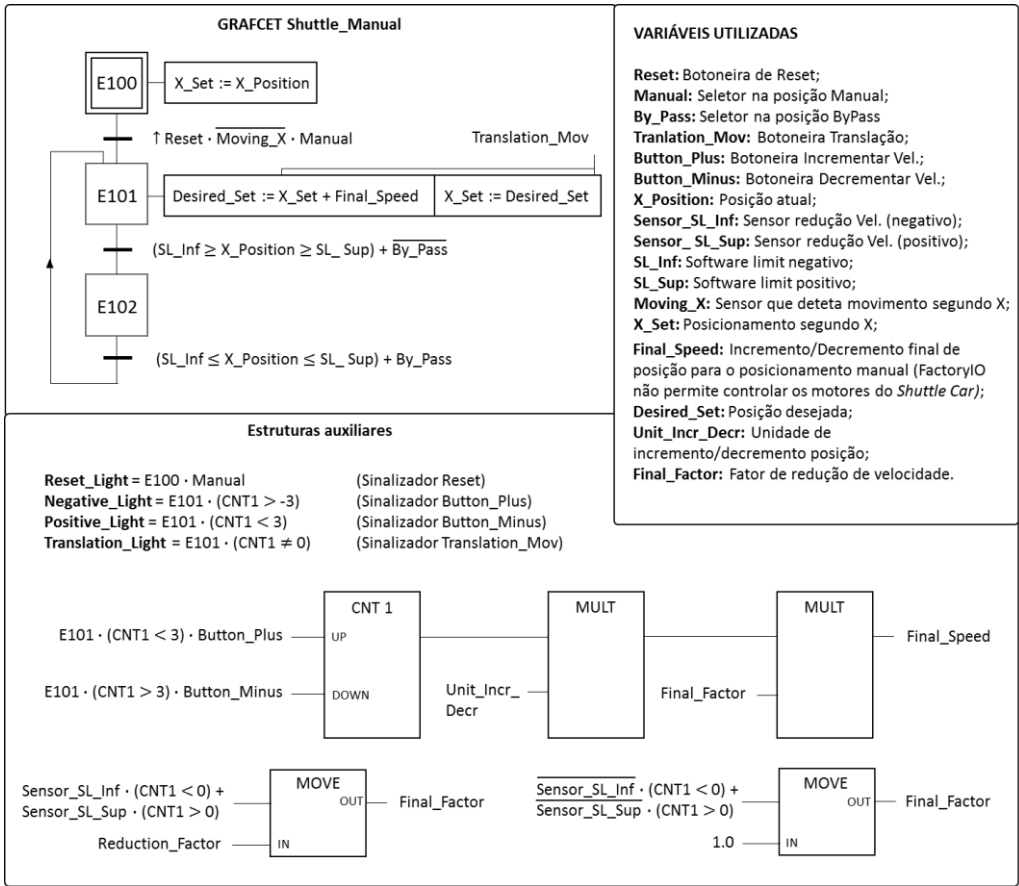


Figura 3 – GRAFCET do *function block* Shuttle_Manual.

Reading_Array

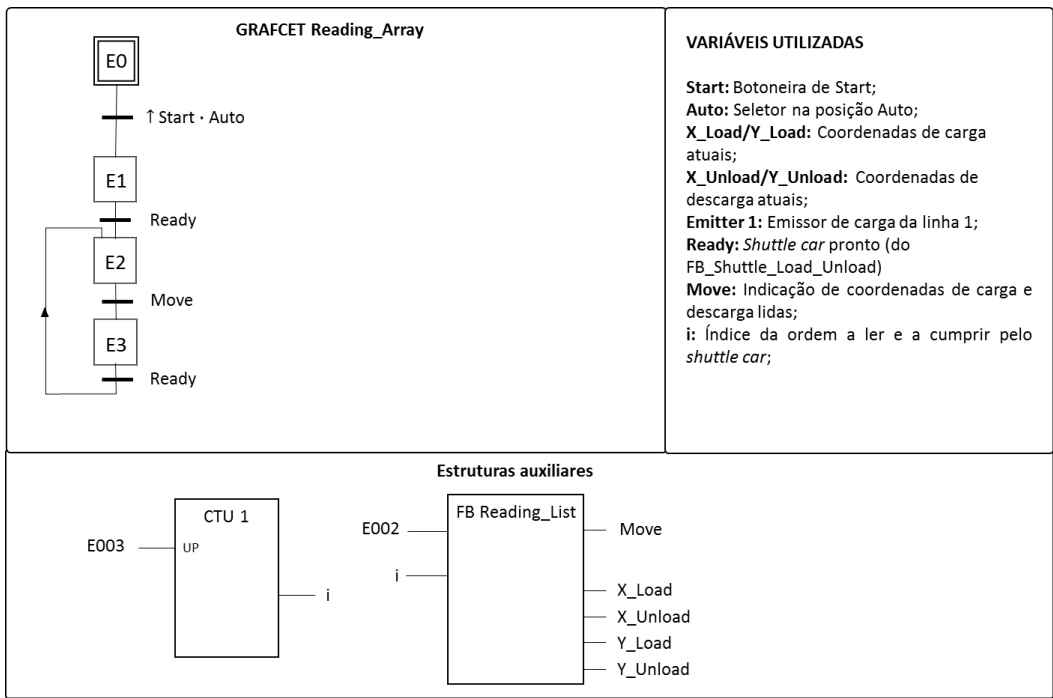


Figura 4 – GRAFCET do *function block* Reading_Array.

#Simulation

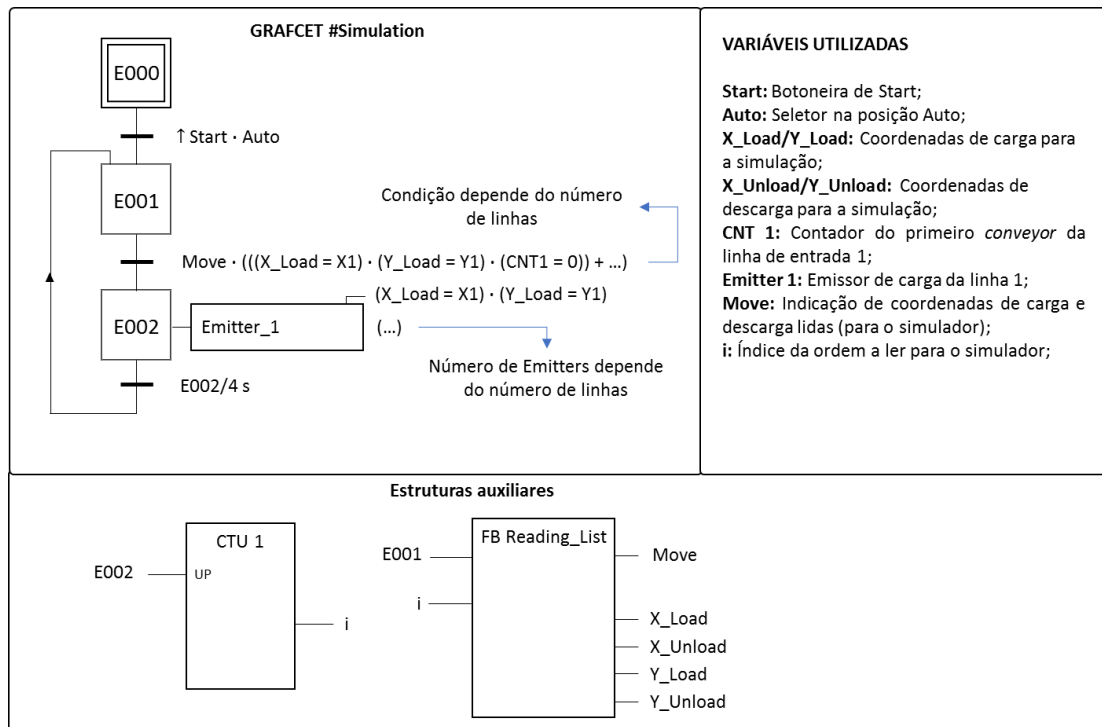


Figura 5 – GRAFCET do *function block* #Simulation.