



# Práticas de Automação Industrial

## Especificação e Programação de Soluções de Controlo Lógico no Ambiente de Treino ITS PLC

**Práticas de automação industrial:  
especificação e programação de soluções de  
controlo discreto no ambiente “ITS PLC”**



Copyright © 2009

Reservados todos os direitos. Proibida a reprodução, total ou parte, por qualquer meio ou processo.

### **Termo de desresponsabilização**

Os problemas e as respectivas soluções apresentados neste livro têm unicamente propósitos didáticos. Em particular, o treino da programação de Controladores Lógicos Programáveis (PLCs) com recurso à plataforma “ITS PLC”. Embora o autor e o editor creiam que as informações apresentadas estão correctas, em caso algum podem ser responsabilizados pela utilização dos programas fornecidos, ou de outros neles baseados, em aplicações das quais possam resultar danos ou prejuízos em pessoas ou bens.

---

# **Práticas de automação industrial: especificação e programação de soluções de controlo discreto no ambiente “ITS PLC”**

---

por

**António Pessoa de Magalhães**

**1ª Edição – 2009**

**Real Games Lda. – [www.realgames.pt](http://www.realgames.pt) – Porto, PORTUGAL**



**Ao**  
**João Rodrigo e ao José Diogo**  
**que tanto gostam de “engenhocas”...**



# Índice

---

---

<b>PREFÁCIO</b> .....	<b>7</b>
<b>PARTE 1 – APRESENTAÇÃO</b> .....	<b>9</b>
<i>O Jogo</i> .....	<i>11</i>
<i>Os Jogadores</i> .....	<i>13</i>
<i>O Equipamento</i> .....	<i>14</i>
<b>PARTE 2 – OS PROBLEMAS</b> .....	<b>17</b>
<b>MISSÃO 1: AUTOMATIZAÇÃO DE UMA ESTAÇÃO DE TRANSPORTE E TRIAGEM DE MERCADORIAS EM PALETES</b> .....	<b>19</b>
<i>Acerca desta Missão</i> .....	<i>20</i>
<i>Tarefa 1: Transporte automático de paletes isoladas no tapete de entrada</i> .....	<i>22</i>
<i>Tarefa 2: Alimentação e transporte automático de paletes isoladas no tapete de entrada</i> .....	<i>23</i>
<i>Tarefa 3: Alimentação e transporte automático de paletes em fila no tapete de entrada</i> .....	<i>24</i>
<i>Tarefa 4: Comando automático da mesa rotativa</i> .....	<i>25</i>
<i>Tarefa 5: Alimentação e transporte automático de paletes do cais de entrada ao elevador da direita</i> .....	<i>26</i>
<i>Tarefa 6: Alimentação e transporte automático de paletes com alternância do elevador de saída</i> .....	<i>27</i>
<i>Tarefa 7: Alimentação e transporte automático de paletes com alternância do elevador de saída, lotação limitada nos tapetes de saída e suporte a situações de indisponibilidade dos elevadores</i> .....	<i>28</i>
<i>Tarefa 8: Alimentação e transporte automático de paletes com triagem por alturas</i> .....	<i>29</i>
<i>Tarefa 9: Transporte automático e otimizado de paletes com triagem por alturas</i> .....	<i>30</i>

<i>Tarefa 10: Encerramento e relançamento da instalação através das botoneiras Iniciar e Parar .....</i>	<b>31</b>
<i>Tarefa 11: Preparação, encerramento e relançamento da instalação através das botoneiras Iniciar e Parar .....</i>	<b>32</b>
<i>Tarefa 12: Produção automática de lotes .....</i>	<b>33</b>
<i>E agora que a primeira missão foi cumprida... ..</i>	<b>34</b>
ITS SUPER – Integração de sistemas de supervisão e terminais de operação .....	<b>35</b>
ITS DEEP – Detecção e suporte de situações de erro.....	<b>36</b>
<b>MISSÃO 2: AUTOMATIZAÇÃO DE UMA ESTAÇÃO DE PRODUÇÃO DE TINTAS .....</b>	<b>37</b>
<i>Acerca desta Missão.....</i>	<b>38</b>
<i>Tarefa 1: Produção de um tanque de tinta vermelha .....</i>	<b>40</b>
<i>Tarefa 2: Produção de uma dose de tinta vermelha.....</i>	<b>41</b>
<i>Tarefa 3: Produção multiciclo de tinta vermelha com parâmetros configuráveis .</i>	<b>42</b>
<i>Tarefa 4: Produção multiciclo de tinta vermelha com dosagem configurável e sinalização de má configuração.....</i>	<b>43</b>
<i>Tarefa 5: Produção otimizada de tinta vermelha .....</i>	<b>44</b>
<i>Tarefa 6: Suporte à paragem antecipada e a procedimentos de alarme .....</i>	<b>45</b>
<i>Tarefa 7: Produção multidoso e multiciclo otimizada de tinta vermelha .....</i>	<b>46</b>
<i>Tarefa 8: Produção flexível e otimizada de cores primárias .....</i>	<b>47</b>
<i>Tarefa 9: Produção por mistura de cores primárias.....</i>	<b>48</b>
<i>Tarefa 10: Produção de tinta por cores tabeladas.....</i>	<b>49</b>
<i>Tarefa 11: Produção de tintas por receitas tabeladas .....</i>	<b>50</b>
<i>Tarefa 12: Produção de tintas por receitas tabeladas com verificação prévia de erros.....</i>	<b>51</b>
<i>E agora que a segunda missão foi cumprida... ..</i>	<b>52</b>
ITS SUPER – Integração de sistemas de supervisão e terminais de operação .....	<b>52</b>
ITS DEEP – Detecção e suporte de situações de erro.....	<b>53</b>
<b>MISSÃO 3: AUTOMATIZAÇÃO DE UM PALETIZADOR ELEVADO .....</b>	<b>55</b>
<i>Acerca desta Missão.....</i>	<b>56</b>
<i>Tarefa 1: Inicialização da máquina .....</i>	<b>58</b>
<i>Tarefa 2: Movimentação cíclica de paletes .....</i>	<b>59</b>
<i>Tarefa 3: Filtragem por software do sinal do sensor 10 .....</i>	<b>60</b>
<i>Tarefa 4: Movimentação de paletes em modo contínuo ou ciclo a ciclo.....</i>	<b>61</b>
<i>Tarefa 5: Comando do alimentador de caixas.....</i>	<b>62</b>
<i>Tarefa 6: Comando sincronizado do alimentador e do acamador .....</i>	<b>63</b>
<i>Tarefa 7: Paletização de uma camada .....</i>	<b>64</b>
<i>Tarefa 8: Paletização de duas camadas .....</i>	<b>65</b>
<i>Tarefa 9: Paletização de três camadas.....</i>	<b>66</b>
<i>Tarefa 10: Paletização flexível por configuração do número de camadas .....</i>	<b>67</b>
<i>Tarefa 11: Paletização flexível por configuração do número de caixas por palete</i>	<b>68</b>
<i>Tarefa 12: Modo de demonstração de paletização flexível .....</i>	<b>69</b>

<i>E agora que a terceira missão foi cumprida.....</i>	<b>70</b>
ITS SUPER – Integração de sistemas de supervisão e terminais de operação .....	<b>70</b>
ITS DEEP – Detecção e suporte de situações de erro .....	<b>71</b>
<b>MISSÃO 4: COMANDO DE UM MANIPULADOR INCREMENTAL EM TAREFAS “PICK AND PLACE” .....</b>	<b>73</b>
<i>Acerca desta Missão .....</i>	<b>74</b>
<i>Tarefa 1: Identificação de peças.....</i>	<b>76</b>
<i>Tarefa 2: Movimentação elementar do manipulador.....</i>	<b>77</b>
<i>Tarefa 3: Comando simultâneo dos dois eixos do manipulador.....</i>	<b>78</b>
<i>Tarefa 4: Comando do manipulador por definição da posição de destino.....</i>	<b>79</b>
<i>Tarefa 5: Inicialização do sistema .....</i>	<b>80</b>
<i>Tarefa 6: Recolha e posicionamento de peças.....</i>	<b>81</b>
<i>Tarefa 7: Suporte ao funcionamento e paragem automáticos.....</i>	<b>82</b>
<i>Tarefa 8: Arrumação de peças por padrões elementares.....</i>	<b>83</b>
<i>Tarefa 9: Suporte a situações de emergência e paragem antecipada.....</i>	<b>84</b>
<i>Tarefa 10: Recolha selectiva de peças e arrumação segundo padrões alternados</i>	<b>85</b>
<i>Tarefa 11: Arrumação de peças por classes.....</i>	<b>86</b>
<i>Tarefa 12: Preenchimento de caixas segundo padrões tabelados.....</i>	<b>87</b>
<i>E agora que a quarta missão foi cumprida... ..</i>	<b>88</b>
ITS SUPER – Integração de sistemas de supervisão e terminais de operação .....	<b>88</b>
ITS DEEP – Detecção e suporte de situações de erro .....	<b>89</b>
<b>MISSÃO 5: ARMAZENAMENTO AUTOMÁTICO DE MERCADORIAS .....</b>	<b>91</b>
<i>Acerca desta Missão .....</i>	<b>92</b>
<i>Tarefa 1: Posicionamento inicial do transelevador.....</i>	<b>94</b>
<i>Tarefa 2: Transferência de mercadorias do transelevador para o casulo 1 e vice-versa .....</i>	<b>95</b>
<i>Tarefa 3: Levantamento de mercadorias armazenadas no casulo 10 .....</i>	<b>96</b>
<i>Tarefa 4: Transferência de mercadorias do cais de entrada para o cais de saída..</i>	<b>97</b>
<i>Tarefa 5: Posicionamento do transelevador por valor de referência .....</i>	<b>98</b>
<i>Tarefa 6: Armazenamento de mercadorias por valor de referência.....</i>	<b>99</b>
<i>Tarefa 7: Armazenamento e levantamento de mercadorias por valor de referência .....</i>	<b>100</b>
<i>Tarefa 8: Armazenamento e levantamento de mercadorias por valor de referência com detecção de erros e suporte a situações de emergência.....</i>	<b>101</b>
<i>Tarefa 9: Armazenamento e levantamento de mercadorias por valor de referência com codificação diversificada .....</i>	<b>102</b>
<i>Tarefa 10: Armazenamento e levantamento de mercadorias por classes.....</i>	<b>103</b>
<i>Tarefa 11: Armazenamento de mercadorias por classes e levantamento por ordem cronológica .....</i>	<b>104</b>
<i>Tarefa 12: Armazenamento e levantamento de mercadorias por valores de referência aleatórios.....</i>	<b>105</b>
<i>E agora que a quinta missão foi cumprida.....</i>	<b>106</b>

ITS SUPER – Integração de sistemas de supervisão e terminais de operação .....	106
ITS DEEP – Detecção e suporte de situações de erro.....	107
<b>PARTE 3 – AS SOLUÇÕES .....</b>	<b>109</b>
<i>O Texto Estruturado e a Norma IEC 61131-3 .....</i>	<i>109</i>
Literatura e Materiais de Apoio .....	111
<i>Acerca das Soluções.....</i>	<i>112</i>
<i>Declaração das Variáveis de I/O .....</i>	<i>113</i>
<b>MISSÃO 1: AUTOMATIZAÇÃO DE UMA ESTAÇÃO DE TRANSPORTE E TRIAGEM DE MERCADORIAS EM PALETES .....</b>	<b>117</b>
<i>Variáveis e instâncias de blocos funcionais usadas na Missão 1 .....</i>	<i>117</i>
<i>Resolução da Tarefa 1 .....</i>	<i>122</i>
<i>Resolução da Tarefa 2 .....</i>	<i>124</i>
<i>Resolução da Tarefa 3 .....</i>	<i>127</i>
<i>Resolução da Tarefa 4 .....</i>	<i>130</i>
<i>Resolução da Tarefa 5 .....</i>	<i>133</i>
<i>Resolução da Tarefa 6 .....</i>	<i>136</i>
<i>Resolução da Tarefa 7 .....</i>	<i>140</i>
<i>Resolução da Tarefa 8 .....</i>	<i>144</i>
<i>Resolução da Tarefa 9 .....</i>	<i>149</i>
<i>Resolução da Tarefa 10 .....</i>	<i>154</i>
<i>Resolução da Tarefa 11 .....</i>	<i>160</i>
<i>Resolução da Tarefa 12 .....</i>	<i>167</i>
<b>MISSÃO 2: AUTOMATIZAÇÃO DE UMA ESTAÇÃO DE PRODUÇÃO DE TINTAS .....</b>	<b>173</b>
<i>Variáveis e instâncias de blocos funcionais usadas na Missão 2 .....</i>	<i>173</i>
<i>Resolução da Tarefa 1 .....</i>	<i>178</i>
<i>Resolução da Tarefa 2 .....</i>	<i>181</i>
<i>Resolução da Tarefa 3 .....</i>	<i>183</i>
<i>Resolução da Tarefa 4 .....</i>	<i>187</i>
<i>Resolução da Tarefa 5 .....</i>	<i>191</i>
<i>Resolução da Tarefa 6 .....</i>	<i>197</i>
<i>Resolução da Tarefa 7 .....</i>	<i>201</i>
<i>Resolução da Tarefa 8 .....</i>	<i>205</i>
<i>Resolução da Tarefa 9 .....</i>	<i>209</i>
<i>Resolução da Tarefa 10 .....</i>	<i>217</i>
<i>Resolução da Tarefa 11 .....</i>	<i>225</i>
<i>Resolução da Tarefa 12 .....</i>	<i>233</i>
<b>MISSÃO 3: AUTOMATIZAÇÃO DE UM PALETIZADOR ELEVADO .....</b>	<b>241</b>
<i>Variáveis e instâncias de blocos funcionais usadas na Missão 3 .....</i>	<i>241</i>
<i>Resolução da Tarefa 1 .....</i>	<i>244</i>

<i>Resolução da Tarefa 2</i> .....	247
<i>Resolução da Tarefa 3</i> .....	250
<i>Resolução da Tarefa 4</i> .....	254
<i>Resolução da Tarefa 5</i> .....	258
<i>Resolução da Tarefa 6</i> .....	262
<i>Resolução da Tarefa 7</i> .....	267
<i>Resolução da Tarefa 8</i> .....	273
<i>Resolução da Tarefa 9</i> .....	279
<i>Resolução da Tarefa 10</i> .....	284
<i>Resolução da Tarefa 11</i> .....	289
<i>Resolução da Tarefa 12</i> .....	295
<b>MISSÃO 4: COMANDO DE UM MANIPULADOR INCREMENTAL EM TAREFAS “PICK AND PLACE”</b> .....	<b>301</b>
<i>Variáveis e instâncias de blocos funcionais usadas na Missão 4</i> .....	<b>301</b>
<i>Resolução da Tarefa 1</i> .....	<b>305</b>
<i>Resolução da Tarefa 2</i> .....	<b>307</b>
<i>Resolução da Tarefa 3</i> .....	<b>309</b>
<i>Resolução da Tarefa 4</i> .....	<b>311</b>
<i>Resolução da Tarefa 5</i> .....	<b>313</b>
<i>Resolução da Tarefa 6</i> .....	<b>317</b>
<i>Resolução da Tarefa 7</i> .....	<b>324</b>
<i>Resolução da Tarefa 8</i> .....	<b>331</b>
<i>Resolução da Tarefa 9</i> .....	<b>337</b>
<i>Resolução da Tarefa 10</i> .....	<b>343</b>
<i>Resolução da Tarefa 11</i> .....	<b>349</b>
<i>Resolução da Tarefa 12</i> .....	<b>356</b>
<b>MISSÃO 5: ARMAZENAMENTO AUTOMÁTICO DE MERCADORIAS</b> .....	<b>365</b>
<i>Variáveis e instâncias de blocos funcionais usadas na Missão 5</i> .....	<b>365</b>
<i>Resolução da Tarefa 1</i> .....	<b>369</b>
<i>Resolução da Tarefa 2</i> .....	<b>371</b>
<i>Resolução da Tarefa 3</i> .....	<b>374</b>
<i>Resolução da Tarefa 4</i> .....	<b>377</b>
<i>Resolução da Tarefa 5</i> .....	<b>380</b>
<i>Resolução da Tarefa 6</i> .....	<b>385</b>
<i>Resolução da Tarefa 7</i> .....	<b>390</b>
<i>Resolução da Tarefa 8</i> .....	<b>396</b>
<i>Resolução da Tarefa 9</i> .....	<b>403</b>
<i>Resolução da Tarefa 10</i> .....	<b>408</b>
<i>Resolução da Tarefa 11</i> .....	<b>413</b>
<i>Resolução da Tarefa 12</i> .....	<b>419</b>
<b>PARTE 4 – EPÍLOGO</b> .....	<b>425</b>



# Prefácio

---

---

Há muitos anos que suporto o ensino da programação de controladores lógicos programáveis (PLCs) em sistemas virtuais. Sem retirarem o devido espaço aos reais, os sistemas virtuais são uma solução de baixo custo, sem riscos para formadores e formandos, de instalação trivial e fáceis de multiplicar. E tudo isto mantendo os formandos em contacto com os equipamentos de controlo de interesse, como se de instalações reais se tratasse.

Mas, quando a estas virtudes se junta uma motivação acrescida por uma simulação extremamente fidedigna e interactiva, que transporta para a sala de aula o ambiente e entusiasmo dos modernos jogos de computador, contagiando formadores e formandos, alcançam-se, então, condições muito favoráveis, quase únicas, a um ensino mais simplificado, célere, natural e eficiente.

Acredito assim que o software de treino “ITS PLC” é um meio privilegiado para alcançar um ambiente de grande entusiasmo, capaz de incutir nos alunos o desejo de realização, pensamento crítico e tenacidade que exibem à frente do ecrã de uma máquina de jogos. Por isso, aceitei de bom grado, com muito orgulho e empenho, o convite que a Real Games Lda. me dirigiu no sentido de produzir um “livro de exercícios” que acompanhasse o seu produto “ITS PLC”.

Foi uma tarefa complexa, mas simultaneamente muito interessante e enriquecedora que, não raras vezes, me fez sentir no ambiente de um jogo de computador. Seguindo precisamente a metodologia destes, elaborei um guião em que cada missão se inicia por pequenos desafios, fundamentais mas facilmente ultrapassáveis, evoluindo depois numa

sequência de exercícios em que cada um acrescenta sempre algo, mas não demasiado, ao anterior, proporcionando assim ao formando ganhos crescentes de conhecimentos e de auto-confiança. Mas, naturalmente que cada guião mais não é do que uma proposta de trabalho, que cada formador poderá, e deverá, adaptar aos seus interesses e formandos.

A apresentação das soluções em texto estruturado será talvez uma surpresa para muitos, dado não ser a linguagem de programação mais comum. Há, contudo, duas fortes razões para tal escolha: por um lado, é uma linguagem de programação textual e de alto nível – logo, muito bem adaptada à comunicação com um público heterogéneo; por outro, contribui fortemente, mas com naturalidade, para a divulgação da norma IEC 61131-3. Quem programa habitualmente PLCs que não seguem esta norma nem empregam esta linguagem, terá certamente sentido pouco, e reflectido ainda menos, sobre as virtudes de ambas. São exactamente esses os leitores que mais se pretende sensibilizar com esta iniciativa.

Reconhecendo que o texto estruturado não é a linguagem de eleição de muitos programadores, é muito possível que as soluções apresentadas venham a ser traduzidas nas mais diversas linguagens e dialectos de programação de PLCs. O interesse de tais traduções é compreensível, sendo até de prever a compilação e disponibilização das mesmas por iniciativa de algum grupo de interesse. Mas, obviamente que as potencialidades do “ITS PLC” não se esgotam nessas traduções e, muito menos, nas propostas contidas neste texto. Haverá sempre espaço para o prazer e o desejo de descobrir novos problemas e novas soluções; ou seja, para novos jogos e formas de jogar o “ITS PLC”. Saibam, formadores e formandos, partilhar esses prazeres.

Porto, Outubro de 2009

António J. Pessoa de Magalhães

---

---

# PARTE 1 – APRESENTAÇÃO

---

---

Este livro propõe um conjunto de exercícios para o ambiente de treino “ITS PLC”. O seu principal propósito é rentabilizar a utilização deste software, sugerindo planos de trabalho capazes de proporcionar uma aprendizagem progressiva, coerente, sólida e aplicada de técnicas de controlo lógico, ou controlo de sistemas de eventos discretos, e correspondente programação de Controladores Lógicos Programáveis (“Programmable Logic Controllers” – PLCs).

Embora as propostas apresentadas sejam naturalmente inspiradas nos ambientes virtuais “ITS PLC”, pretende-se que delas resultem ensinamentos cujo interesse e aplicação as transcendam o mais possível. Assim, se no seu lado mais vistoso e abrangente, as instalações “ITS PLC” servem de mote a problemas lógicos relativamente latos e um tanto avançados na esfera da automação industrial, na sua vertente mais prática servem de pretexto para introduzir e discutir questões pontuais e elementares que vão ao encontro dos erros, dúvidas e dificuldades mais comuns de quem se inicia, ou pretende evoluir, na programação de PLCs. Para a definição, apresentação, tratamento e encadeamento lógico destes problemas, em muito contribuiu a experiência relativamente longa do autor no ensino da programação de PLCs, uma actividade sempre apoiada tanto em sistemas reais como virtuais.

Conquanto o principal desafio do conjunto de exercícios propostos seja programar um PLC para controlar correcta e elegantemente cada instalação “ITS PLC”, excelente é que tal feito seja, simultaneamente, um ponto de partida para o desenvolvimento de aplicações mais abrangentes. Por exemplo, aplicações que integrem consolas de interface homem-máquina, sistemas de supervisão, controlo distribuído ou gestão de informação. Extremamente útil é também procurar soluções robustas e capazes de lidar com situações de falha ou insegurança. Propostas de trabalho orientadas em ambos sentidos são incluídas neste texto. Têm em mente níveis de educação e treino mais elevados, devendo por isso os formadores adaptar tais linhas orientadoras aos

objectivos dos seus cursos, equipamentos disponíveis e níveis de conhecimentos dos seus formandos.

Considerando unicamente os exercícios que têm como pano de fundo o controlo por PLC das cinco instalações virtuais “ITS PLC”, este livro propõe um total de sessenta problemas: doze por cada instalação. Para cada problema proposto é fornecida a respectiva solução na forma de um programa de PLC, devidamente comentado. Nos casos mais simples, os programas são antecidos de uma explicação relativamente curta e informal; nos mais complexos, a solução é fundamentada numa especificação em linguagem *GRAF CET*<sup>1</sup> e de acordo a norma IEC 60484, segunda edição. Longe de serem as únicas possíveis, as especificações e as soluções apresentadas têm o propósito de serem modulares, genéricas e, desejavelmente, potenciadoras de inúmeros motivos de reflexão por parte de formadores e formandos.

É cada vez mais comum começar o projecto de um sistema sequencial pela sua especificação em *GRAF CET*. O facto de o *GRAF CET*, por vezes erradamente designado “Sequential Function Chart”, ser uma metodologia gráfica normalizada muito mais sucinta, objectiva e abrangente do que os diagramas de estado e os diagramas temporais, tem-no tornado numa ferramenta amplamente usada e bastante bem conhecida da generalidade dos programadores de PLCs. A tudo isso acresce ainda a relativa simplicidade do processo de tradução de um *grafcet* num programa genérico de PLC, uma questão que merece também o devido destaque neste livro. Consequentemente, adoptar neste texto o *GRAF CET* como linguagem descritiva do comportamento de um sistema não mereceu qualquer hesitação.

Já a escolha da linguagem de programação em que as resoluções são apresentadas mereceu grande ponderação. Acabou por recair sobre o “texto estruturado”, tal como semântica e funcionalmente previsto na norma IEC 61131-3, segunda edição. Para além de bem adaptada ao fim em vista, esta opção promove uma linguagem e uma norma de crescente importância no domínio dos PLCs, alargando, deste modo, o domínio didáctico deste texto.

Porque se reconhece que o texto estruturado e a norma IEC 61131-3 nem sempre são bem conhecidos de quem se inicia na programação de PLCs, houve a preocupação de incluir neste livro um conjunto de informações orientadoras dos programadores menos familiarizados com esta linguagem e com a norma em causa. Essa informação precede a apresentação das soluções. E, falando em normas, interessante será também encontrar nos exercícios propostos espaço para promover a norma IEC 61499. Embora tal meta saia um pouco dos objectivos primários deste texto, os formadores mais familiarizados com o tema saberão certamente encontrar o caminho certo para a alcançar.

---

<sup>1</sup>Seguindo a tendência da literatura especializada, o texto adopta o termo *GRAF CET* para designar a linguagem de especificação “GRAPhe Fonctionnel de Commande Étape Transition” e o vocábulo *grafcet* para designar um esquema gráfico que utiliza a linguagem *GRAF CET*.

Acredita-se, pois, que há inúmeros motivos de interesse neste livro, o qual está dividido em quatro partes organizadas do seguinte modo:

A Parte 1 – Apresentação – expõe as questões necessárias a uma correcta e completa introdução do leitor ao ambiente de aprendizagem que o espera. Concretamente, começa por apresentar o enquadramento, a sequência, os objectivos e o público-alvo dos desafios lançados neste texto. Seguidamente, são feitas algumas considerações sobre o modo como o leitor deve encarar os desafios que lhe serão lançados e a sua previsível aprendizagem face aos seus conhecimentos prévios. Por fim, são listados os recursos necessários à realização e acompanhamento dos exercícios propostos.

A Parte 2 – Os Problemas – apresenta, como se de um jogo de computador se tratasse, os cinco grandes desafios, ou “missões”, que o leitor deverá progressivamente resolver. Cada desafio corresponde ao controlo de uma instalação virtual “ITS PLC” e está organizado em doze exercícios de programação. Estes estão sequenciados em pequenas tarefas, de forma a potenciarem uma aprendizagem simples, natural e eficiente. No início de cada missão são dadas explicações sobre o ambiente em que a mesma se desenrola, de modo que o leitor compreenda minimamente os problemas típicos da instalação industrial emulada e o interesse prático dos exercícios propostos. No fim de cada missão são lançadas propostas de trabalho destinadas a um público com conhecimentos mais avançados. Estas propostas estão organizadas segundo duas perspectivas: por um lado, a integração das aplicações com sistemas de supervisão e consolas de interface homem-máquina; por outro, melhorar a robustez dos programas desenvolvidos por inclusão de mecanismos lógicos de detecção e suporte de situações anómalas.

A Parte 3 – As Soluções – inicia-se com algumas considerações sobre a programação em texto estruturado e a norma IEC 61131-3. Esta introdução tem em mente os leitores menos familiarizados com estes temas, sugerindo-lhes também literatura, sítios na Internet e recursos de software julgados úteis. Feita essa breve apresentação, surgem então as soluções dos desafios lançados na Parte 2. Cada solução compreende a justificação dos procedimentos a programar, por vezes apoiada num *grafcet*, seguida do programa correspondente, devidamente comentado.

A Parte 4 – Epílogo – encerra o texto com um conjunto sumário de conclusões.

## O Jogo

Bem-vindo ao jogo “ITS PLC”! Se não encontrou este livro absolutamente por acaso, sabe que o “ITS PLC” é um pacote de software desenvolvido pela Real Games Lda. que

emula instalações industriais a controlar por PLC. Informações sobre este produto estão disponíveis no sítio [www.realgames.pt](http://www.realgames.pt).

É pois importante que, antes de continuar esta leitura, se familiarize minimamente com o software em causa, compreendendo a sua finalidade e potencialidades, assim como as características e os meios de comando e de sensorização das instalações emuladas. Uma versão não licenciada serve para este primeiro contacto. Pode obtê-la no sítio da Real Games Lda., assim como o respectivo manual de utilização.

O principal objectivo deste texto é ajudá-lo a “dar vida” a cada um dos cinco cenários que compõem o “ITS PLC”. Para isso, o leitor, que é como quem diz, “o jogador”, terá de interligar um PLC ao computador onde tem instalado o “ITS PLC” e programá-lo correctamente.

Conhecendo a aplicação, facilmente o leitor depreenderá que este jogo lhe propõe cinco missões. Em lato senso, elas são:

- Automatizar uma instalação de “Sorting”;
- Automatizar uma instalação de “Batching”;
- Automatizar um Paletizador;
- Automatizar uma instalação “Pick and Place”;
- Automatizar um Armazém Automático.

O interesse prático e os objectivos exactos de cada missão serão revelados oportunamente. Para já, importa dizer que cada missão é composta por um conjunto de doze tarefas, a realizar pela ordem proposta, de modo que o jogador se familiarize progressivamente com o sistema a automatizar e adquira as competências necessárias para o completo e efectivo cumprimento da missão atribuída. Cumprir uma tarefa significa subir um nível. Vencer o último nível significa completar a missão. Completar as cinco missões significa terminar o jogo e ser um “perito em programação de PLCs”!

A justificação e demonstração do interesse prático de cada missão são aspectos a que foi dado um particular cuidado. Assim, cada missão inicia-se com uma explicação dos aspectos físicos e funcionais da instalação emulada, permitindo compreender devidamente o interesse, os objectivos e as dificuldades da missão em causa no contexto de instalações e aplicações reais congéneres.

Cada tarefa tem um enunciado muito simples, e preciso, que inclui o cenário em que se desenrola, o objectivo a atingir e os sinais de I/O a considerar. Algumas tarefas visam o comando automático de apenas parte do equipamento disponível, tendo o jogador de comandar manualmente outras partes. Nesses casos, os enunciados explicam como e porquê. O mesmo acontece quando a verificação da correcção dos programas

desenvolvidos requer a simulação de avarias ou a imposição de um estado funcional a um ou mais actuadores.

O jogador dispõe de duas “pistas” para realizar cada tarefa. Utilizá-las faz naturalmente sentido. Mas começar por procurar as suas próprias pistas faz bastante mais...

Para cada tarefa proposta há uma resolução disponível. Concluída uma tarefa, o jogador deverá comparar a sua solução com a fornecida. Se acabar por concluir que, definitivamente, não é capaz de realizar uma tarefa, deve então consultar e estudar cuidadosamente a solução proposta, procurando entendê-la completamente antes de passar à tarefa seguinte. Mas é importante que não desista cedo demais! E, tanto em caso de sucesso como de insucesso, deve meditar nos comentários e justificações que acompanham a solução proposta.

Completada uma missão, isto é, terminada a automatização de um sistema, há ainda espaço para alargar os horizontes da mesma. Nesse sentido, o jogador é desafiado para metas mais ambiciosas enunciadas em dois “pacotes” de exercícios suplementares:

- ITS SUPER – SUPERvisory Environments and Systems e
- ITS DEEP – DEpendable Environments Programming.

O primeiro visa o desenvolvimento de soluções flexíveis e distribuídas por integração de recursos tecnológicos – tais como PLCs, consolas HMI, sistemas de supervisão e outros. O segundo encerra um conjunto de desafios destinados a melhorar a confiabilidade das soluções encontradas por inclusão de técnicas de detecção e suporte de situações anómalas. Mais do que exercícios muito específicos, com enunciados concretos e rígidos, as propostas contidas nestes dois “pacotes” são essencialmente linhas orientadoras de trabalhos mais latos, de nível mais elevado e que exigem mais tempo. Cabe por isso aos formadores adaptá-las aos equipamentos ao seu alcance e aos interesses e conhecimentos dos seus formandos. Uma possibilidade interessante é considerá-los como tema daqueles pequenos projectos de desenvolvimento, individuais ou em grupo, que habitualmente são propostos aos formandos de níveis mais avançados.

## **Os Jogadores**

O software de treino “ITS PLC” é uma plataforma didáctica de programação de PLCs que, quando limitada a problemas simples, pode e deve ser utilizada por formandos que dão os primeiros passos nesta matéria. Já programar um PLC para comandar e dotar os cinco sistemas incluídos no pacote “ITS PLC” de funcionalidades um tanto elaboradas, é uma

tarefa que exige um leque relativamente amplo de conhecimentos e competências. Concretamente, nas seguintes matérias:

- Sistemas lógicos: variáveis binárias e códigos binários; sistema de numeração binário, octal e hexadecimal; operadores lógicos e álgebra de Boole; elementos de memória; operações de “set” e “reset”; operações sobre registos; operações aritméticas, de rotação e de deslocamento;
- Especificação: representação funcional em diagramas temporais, diagramas de estado e *GRAF CET*;
- PLCs: modelo de funcionamento de um PLC; afectação e cablagem de entradas e saídas; registos e organização interna da memória; temporizadores e contadores; experiência mínima em programação de PLCs e na utilização de ferramentas de desenvolvimento e teste associadas; facilidade em ler e compreender o manual de um PLC.

Se tudo isto lhe é familiar, então considere-se apto a começar o jogo e levá-lo até ao fim. Se, pelo contrário, não conhece ou domina minimamente grande parte destes temas, então deve procurar melhorar um pouco os seus conhecimentos antes de aceitar os desafios que aqui se lançam. Há um rol infindável de literatura sobre estes assuntos. O seu formador saberá, seguramente, indicar-lhe a bibliografia mais adequada aos seus conhecimentos.

Para quem se prepara para começar a jogar, o conselho mais óbvio é o de que, tal como nos jogos de computador, “evite a batota”. Ter dificuldades em encontrar uma solução, e tê-la ao virar da página, pode não ser o melhor estímulo à perseverança. Mas, sempre que tiver a tentação de espreitar a resolução, lembre-se que perder a oportunidade de descobrir a sua própria solução – que, quem sabe, seria até mais interessante do que a proposta – não é o melhor contributo para uma boa aprendizagem.

Procure pois encontrar sempre as suas próprias soluções. Verificará que, desse modo, os seus conhecimentos evoluem e solidificam de uma forma muito natural e irreversível. Se optar por espreitar sistematicamente a solução sentirá que, na maioria dos casos, pouco tempo depois, já não se “lembrará do truque da solução”, concluindo que, afinal, não aprendeu tanto quanto desejaria. Talvez nem tenha compreendido que a programação de PLCs vive da “lógica” e não de “truques”!

## O Equipamento

Certamente tem a noção exacta do equipamento que necessita para começar o jogo: um computador com o “ITS PLC” instalado, e devidamente licenciado, e um PLC com um

número mínimo de entradas e saídas. Verifique que tipo de entradas e saídas tem o seu PLC e certifique-se que as ligou correctamente à placa de I/O que acompanha o produto. Consulte o manual de utilização do “ITS PLC” sobre esta questão. Tenha, aliás, este manual sempre por perto, pois vai precisar de consultar frequentemente o mapa de entradas e saídas das instalações a controlar.

Por perto deverá ter também o manual do seu PLC e do software de desenvolvimento associado. A ajuda em linha nem sempre é suficiente. Ter também à mão um ou outro livro sobre sistemas lógicos e especificação em *GRAFCET* (de preferência, a segunda edição da norma IEC 60484) é também uma boa ideia.

Mesmo um PLC relativamente modesto tem capacidade para controlar as instalações apresentadas. Mas, se tem um PLC relativamente sofisticado, não deixe de explorar as suas potencialidades, inventando novos exercícios ou outras formas de os resolver que as empreguem.

Se tem possibilidade de ligar dois monitores ao seu computador, deve fazê-lo. Use um para visualizar o ambiente “ITS PLC” e outro para analisar “on-line” toda a informação relativa à execução do programa no PLC recorrendo a uma ferramenta de “debugging”. Rapidamente compreenderá os benefícios desta estratégia.

Por último, se é uma pessoa que gosta de jogos de computador e se entusiasma facilmente com problemas lógicos, aconselho-o a fazer-se acompanhar de uma garrafa térmica com o seu chazinho preferido. Talvez o dia, ou a noite, seja longo(a)...

Exposto o essencial sobre o ambiente do jogo “ITS PLC”, é tempo de avançar para a máquina de jogos!



---

---

## PARTE 2 – OS PROBLEMAS

---

---

Provavelmente está já à frente do seu computador, devidamente equipado e pronto a conhecer o primeiro desafio. Algumas breves notas antes de começar:

Conforme referido, os problemas são propostos em cinco módulos, auto-contidos, seguindo a ordem por que as instalações são apresentadas no software “ITS PLC”: “sorting”, “batching”, “paletizador”, “pick and place” e “armazém automático”. As missões não têm necessariamente de ser realizadas por esta ordem, mas aconselha-se a que o sejam. Isso porque, no sentido de evitar a duplicação de explicações detalhadas, a resolução de algumas tarefas remete para explicações prestadas em soluções anteriores. Mas, importante mesmo, é que, dentro de cada missão, tente resolver as tarefas pela ordem por que são apresentadas, uma vez que, no contexto da missão, cada tarefa complementa a anterior e será complementada pela seguinte. Quando achar que está a ser difícil evoluir numa missão, sugere-se, como primeiro passo, que a abandone temporariamente, sem consultar a solução, e passe à missão seguinte. Talvez esta lhe proporcione os ensinamentos e a inspiração que lhe permitirão retomar, mais tarde, a missão que deixou em suspenso.

Importante também é que se certifique sempre que compreendeu bem a questão que lhe é posta antes de pensar na respectiva resolução. Verifique também que o I/O indicado no enunciado é coerente com os objectivos da tarefa. Comece sempre por definir minimamente, mas de forma objectiva, os procedimentos a realizar. O recurso a um diagrama temporal, diagrama de estados ou *grafcet* é uma boa abordagem, especialmente nos casos mais complexos. Só depois se preocupe com o programa a desenvolver. Se o enunciado de algum problema não lhe parecer absolutamente claro, não use tal argumento para consultar as soluções. Procure resolver a questão tomando a interpretação que lhe parecer mais plausível, e não a mais simples!

Por fim, dois conselhos práticos:

Ao longo das suas experiências vai certamente ter, por vezes, interesse em reiniciar o seu sistema; isto é, tanto a instalação virtual como o seu PLC. A instalação é limpa premindo o botão “Limpar” no painel do “ITS PLC”. Para reiniciar o PLC, sugere-se a seguinte dica: inclua no seu programa um procedimento para reinicializar as variáveis internas do PLC quando uma entrada não utilizada, por exemplo, um sensor ou uma botoneira, é forçada ou accionada. Pode, desse modo, e sempre que quiser, concretamente depois de premir “Limpar”, reiniciar o seu PLC sem o retirar do modo “run”. Pode, também, optar por utilizar o selector “Manual/Auto” (Entrada 11) para o mesmo fim. Mas, nesse caso, o PLC reiniciará sempre que a instalação é lançada no modo automático. Tal situação nem sempre é a mais interessante.

Em qualquer caso, é conveniente que não coloque o seu PLC em modo “run” antes de colocar a instalação em modo automático, pois, nessa situação, as variáveis internas do programa do PLC podem evoluir para valores diferentes dos pretendidos para a situação inicial da instalação.

Feitas estas considerações, chegou finalmente o momento de conhecer os desafios que o esperam. Boa Sorte!

---

## Missão 1: Automatização de uma estação de transporte e triagem de mercadorias em paletes

---

**OBJECTIVO:** ENCAMINHAR PALETES DO CAIS DE ENTRADA AOS ELEVADORES DE SAÍDA, SELECIONANDO-AS POR ALTURAS



## Acerca desta Missão

O movimento de bens em transportadores automáticos, tais como tapetes rolantes e mesas de transferência, é uma tarefa muito comum em instalações industriais. Do ponto de vista funcional, um tapete rolante com um só sentido de movimentação é o elemento de transporte automático mais simples, podendo o seu estado, movimenta ou não mercadoria, ser representado por uma variável binária. Há, contudo, transportadores muito flexíveis, relativamente complexos, cujo leque de estados possíveis é obviamente muito mais vasto. É o caso dos transportadores sequenciais, como a mesa rotativa desta aplicação, que, servindo simultaneamente de elementos de transferência e triagem, têm até, por vezes, diversos pontos de entrada ou saída.

Um transportador tem à sua entrada algo que lhe fornece mercadoria; por exemplo, outro tapete, um operador ou um alimentador automático. À sua saída haverá algo que recebe mercadoria: um outro tapete, uma mesa de transferência, um cais de saída ou um operador. A missão de um transportador é transferir mercadorias de um ponto de partida a um ponto de chegada de forma eficiente. Eficiência pressupõe, tipicamente, um transporte de acordo com uma origem e um destino pré-definidos e realizado no menor tempo e com o menor consumo possível de energia. Ou seja, requer que:

- Um tapete não esteja em movimento caso nele não exista qualquer mercadoria;
- Um tapete que tenha mercadoria não esteja parado, a menos que tal seja absolutamente necessário;
- Os elementos de triagem encaminhem as mercadorias para os destinos correctos.

O principal objectivo desta missão é mostrar que, mesmo numa instalação complexa, o que de facto existe à entrada e à saída de um transportador pouco importa para o seu comando: importante é a sincronização do funcionamento de cada tapete com o dos alimentadores e transportadores que tem à sua entrada e à sua saída. É dessa compreensão que resulta a capacidade para desenvolver uma solução modular, apoiada num controlador centralizado ou distribuído, e perfeitamente utilizável em sistemas de transporte automático reais e de grande dimensão.

Particularmente importante em qualquer sistema de transporte flexível é conseguir gerir toda a informação necessária ao correcto encaminhamento das mercadorias. Para tal, a informação das mercadorias em trânsito, independentemente de ter origem em sistemas de identificação mais ou menos sofisticados, como leitores de códigos de barras ou de RFIDs, ou em vulgares sensores de proximidade, como no caso da presente aplicação, tem normalmente de seguir percursos e passar por processos de seriação e triagem muito semelhantes aos das próprias mercadorias.

Também importante num sistema de transporte é a abertura dos controladores locais à troca de informação com elementos de diálogo homem-máquina e sistemas de supervisão. Por último, mas também de importância vital, há a questão da detecção e suporte eficaz de situações erróneas que possam conduzir à degradação física da instalação ou dos bens transportados.

Esta missão toca em tudo isto, começando pelas coisas mais simples.

## Tarefa 1: Transporte automático de paletes isoladas no tapete de entrada

<b>Cenário:</b>	O tapete de entrada entra em movimento quando lhe chega uma paleta, transporta-a até à mesa rotativa e pára.
<b>Objectivo:</b>	Comandar automaticamente o tapete de entrada para transportar paletes isoladas.
<b>Estado Inicial:</b>	Tapete de entrada sem paletes.
<b>Sinais de I/O:</b>	Entradas: Sensores 0 e 3. Saídas: Actuador 1.
<b>Procedimentos Manuais:</b>	Comandar manualmente o alimentador, forçando e libertando o actuador 0, de modo a fazer chegar uma paleta ao tapete de entrada somente quando este está vazio. Manter a mesa rotativa em carga, forçando o actuador 2, para dar saída às paletes provenientes do tapete de entrada, lançando-as para o chão. Remover caixas e paletes que se acumulem atrás da mesa rotativa, mantendo-a desobstruída.
<b>Dicas:</b>	Use uma variável binária, i.e., um “bit” de memória, para definir constantemente se o tapete deve estar ou não em movimento... A entrada e saída de paletes no tapete correspondem a transições lógicas em sensores...

## Tarefa 2: Alimentação e transporte automático de paletes isoladas no tapete de entrada

<b>Cenário:</b>	O tapete de entrada funciona como na Tarefa 1. O alimentador é agora comandado automaticamente para só lançar uma paleta no tapete de entrada quando este está vazio.
<b>Objectivo:</b>	Automatizar o alimentador, eliminando o comando manual da Tarefa 1.
<b>Estado Inicial:</b>	Tapete de entrada sem paletes.
<b>Sinais de I/O:</b>	Entradas: Sensores 0 e 3 – ou outros, se achar mais conveniente. Saídas: Actuadores 0 e 1.
<b>Procedimentos Manuais:</b>	Manter a mesa rotativa em carga, forçando o actuador 2, para dar saída às paletes provenientes do tapete de entrada, lançando-as para o chão. Remover caixas e paletes que se acumulem atrás da mesa rotativa, mantendo-a desobstruída. Forçar avarias ocasionais no alimentador, impedindo-o de fazer chegar paletes ao tapete de entrada, para verificar que este último efectivamente pára quando não transporta qualquer paleta.
<b>Dicas:</b>	Note que a transferência de uma paleta requer o movimento simultâneo do tapete de entrada e do alimentador... Note que quando o tapete de entrada transporta uma paleta, tal não implica necessariamente que o alimentador deva estar parado. Defina uma variável binária, “Busy_1”, que seja verdadeira quando o tapete de entrada não está em condições de receber uma paleta...

## E agora que a primeira missão foi cumprida...

Parabéns por ter cumprido esta missão! É agora um perito em sistemas de transporte automáticos e, certamente, não teria dificuldades de maior em realizar programas capazes de cobrir as seguintes alterações à instalação, pressupondo a existência dos sensores e actuadores necessários, que deverá ser também capaz de listar com toda a facilidade:

- O transporte entre o alimentador e a mesa rotativa é agora mais longo e passou a exigir dois tapetes, um a seguir ao outro;
- A mesa rotativa tem agora dois tapetes de entrada, com o novo colocado atrás dela. Recebe paletes altas e baixas de ambos, identificadas por dois sensores de proximidade colocados à entrada de cada tapete, e encaminha-as para os elevadores de saída em função das respectivas alturas;
- Passaram a existir paletes com três alturas, que continuam a ser identificadas por sensores de proximidade colocados a cotas apropriadas no início do tapete de entrada. As paletes devem ser devidamente encaminhadas para três elevadores de saída. Para isso, o tapete da esquerda, alimenta agora uma segunda mesa rotativa, a qual pode transferir paletes para dois tapetes de saída que conduzem as paletes para outros tantos elevadores.

Dominadas as questões funcionais da instalação, é interessante pensar-se em acrescentar-lhe flexibilidade e confiança no funcionamento, dois requisitos de extrema importância prática. Se estas questões lhe interessam, então atente nas propostas contidas nos dois pacotes de exercícios suplementares desta missão, o “ITS SUPER” e o “ITS DEEP”, antes de passar à segunda missão.

Cada um destes pacotes lança um conjunto de questões com objectivos relativamente simples, de contornos flexíveis e que, no seu conjunto, podem ser encarados como “um projecto” ou “um trabalho de fundo”, no sentido em que requerem a dedução, programação e avaliação de um leque considerável de procedimentos lógicos. São desafios que exigem algum tempo e recursos, destinados a um público com conhecimentos e interesses que transcendem a programação de PLCs, podendo, inclusivamente, servir de tema àqueles pequenos projectos de desenvolvimento que os formadores habitualmente propõem aos seus formandos de níveis mais avançados, sejam estes desenvolvidos individualmente ou em grupo.

Dado o âmbito e a abrangência das propostas, cabe pois ao leitor, caso os temas lhe interessem, tenha conhecimentos para isso e equipamentos laboratoriais disponíveis, reflectir sobre a melhor solução para os desafios “ITS SUPER” e “ITS DEEP” desta e das demais missões. O ideal é que o faça no círculo do seu grupo habitual de trabalho.

## **ITS SUPER – Integração de sistemas de supervisão e terminais de operação**

O objectivo deste pacote é aumentar a flexibilidade da solução alcançada através de trocas de informação entre o controlador da instalação, i.e., o PLC, e sistemas de supervisão e interface homem-máquina. Se tem uma HMI que possa ligar ao seu PLC, ou um SCADA com que este possa comunicar, nem que seja um pacote de software de demonstração, há que lhes dar uso! Programe o seu SCADA ou HMI para recolher do PLC a informação a seguir listada:

- Estado da Instalação: “standby”, pronta, alarme, etc;
- Estado de cada tapete: inactivo, transporta “*n*” paletes, bloqueado, etc;
- Estado da mesa rotativa: pronta, carga, etc;
- Alturas das paletes no tapete de entrada por ordem cronológica;
- Dados da produção:
  - Indicação de produção contínua ou por lotes;
  - No caso de produção por lotes: extensão, volume já produzido e volume a produzir;
  - Número de paletes, baixas e altas, entradas na instalação durante o trabalho actual;
  - Número de paletes, baixas e altas, já enviadas;
  - Número de paletes, baixas e altas, em trânsito;
  - Se tiver recursos para tal, hora de início do último trabalho e tempo total de funcionamento da instalação nas últimas 24 horas.

Programe também o seu sistema de modo que o PLC possa receber de uma HMI ou de um SCADA os seguintes comandos e configurações:

- Ordens de arranque, limpeza, encerramento e relançamento, incluindo fazer o sistema arrancar e terminar a horas predefinidas ou terminar ao fim de um determinado tempo;
- Configuração da produção: produção contínua ou por lotes com definição dos respectivos parâmetros, ou seja, quantidades e tipos de paletes;
- Configuração das políticas de encaminhamento das paletes: direita, esquerda, alternado, triagem por alturas, etc;
- Definição da lotação do tapete de entrada.

## **ITS DEEP – Detecção e suporte de situações de erro**

O propósito deste pacote é complementar a programação do PLC com funcionalidades tendentes à detecção, suporte e diagnóstico de situações de erro. As situações de erro mais graves, não sendo possíveis de suportar de outra forma, devem levar à paragem do sistema, colocando-o em modo de avaria ou outro julgado mais conveniente – “standby”, por exemplo. Todas as situações de erro e maus funcionamentos detectados devem ser identificados e assinalados através de um “código de erro”. Caso tenha achado o pacote “ITS SUPER” interessante, pode ainda complementá-lo com a comunicação ao SCADA ou a afixação na HMI dos códigos de erros detectados.

Propõe-se, então, que o leitor desenvolva pequenas funções, tão eficientes quanto o possível, para cobrir as situações indesejáveis a seguir listadas, evitando assim o funcionamento da instalação em situações de risco. A eficiência destes mecanismos mede-se por dois parâmetros: a relação entre o número de casos detectados e o número de casos ocorridos – que se denomina “cobertura” – e a quantidade de detecções e intervenções erróneas – que se denomina “falsos alarmes”. O primeiro deve aproximar-se de 100%; o segundo, de 0.

Propõe-se a seguinte lista de eventos a detectar:

- Há duas ou mais paletes encostadas umas às outras à entrada da mesa rotativa;
- Deslocamento da paleta durante a rotação da mesa;
- Perda, por encravamento ou qualquer outro motivo, ou surgimento inesperado de uma paleta num transportador;
- Avaria em qualquer sensor e identificação da mesma – saída sempre a 0 ou sempre a 1;
- Avaria em qualquer actuador e identificação da mesma – encravado a 0 ou a 1.

Utilize as possibilidades de interacção do software e de geração de avarias para testar os seus mecanismos de detecção de erros e avalie as técnicas desenvolvidas.

---

---

## PARTE 3 – AS SOLUÇÕES

---

---

Muito provavelmente, o secreto desejo do leitor, de que as soluções lhe fossem fornecidas na linguagem de programação do seu PLC, não se concretizará. Mas, com tantas marcas e modelos de PLCs, a probabilidade de tal acontecer era, de facto, muito baixa!

Na impossibilidade de apresentar as soluções na linguagem preferida de cada utilizador, optou-se por fazê-lo numa linguagem muito eficiente, de crescente aceitação, facilmente compreensível mesmo para quem jamais contactou com ela, facilmente traduzível noutras linguagens de programação de PLCs e extremamente expressiva e reveladora dos algoritmos de controlo empregues: o Texto Estruturado (“Structured Text” – ST).

### **O Texto Estruturado e a Norma IEC 61131-3**

O texto estruturado é uma linguagem de programação de PLCs moderna e contemplada na norma IEC 61131-3. A sua simplicidade e expressividade têm-lhe granjeado um número sempre crescente de adeptos entre programadores e fabricantes de PLCs.

Se é a primeira vez que contacta com esta linguagem, considere tal facto como mais um ponto de interesse deste livro de exercícios. Verá que em pouco tempo será mais um fã do texto estruturado. E, se conhece minimamente as vulgares linguagens de programação de alto nível, como o BASIC, Pascal ou C, vai constatar que o texto estruturado é, afinal, uma linguagem muito fácil de compreender e utilizar e, de certa forma, já sua conhecida.

Para muitos autores, o texto estruturado é a linguagem que permite uma programação mais rápida e eficiente dos modernos sistemas de controlo baseados em PLC. Conquanto tal seja obviamente discutível, a verdade é que os benefícios desta linguagem relativamente às mais clássicas são particularmente notórios em aplicações complexas. Mas um outro factor tem contribuído em muito para a eficiente concepção, realização e reengenharia de soluções baseadas em PLC: a norma IEC 61131. Se não conhece esta norma é tempo de a conhecer. Se trabalha ou estuda numa Universidade, encontrá-la-á certamente na biblioteca da sua instituição. Se não é o caso, a consulta de alguns dos inúmeros livros, artigos e sítios na Internet que dedicam grande atenção a esta norma, e em especial à Parte 3 (IEC 61131-3), podê-lo-ão ajudar. Apresentar as soluções dos exercícios propostos em texto estruturado serve assim também de pretexto para incitar o leitor a conhecer a norma IEC 61131, pese embora muitos dos seus aspectos fundamentais não se reflectam nas resoluções propostas.

Mas se é sua intenção continuar a utilizar uma linguagem de programação que “nada tem a ver com o texto estruturado” – esquemas de contactos, diagramas de blocos, lista de instruções – então, a norma IEC 61131-3 também é para si, já que ela contempla igualmente estas linguagens. Qualquer uma delas podia, por isso, ter sido utilizada para apresentar a resolução dos exercícios propostos, mantendo-se os mesmos propósitos de divulgação e conformidade com a norma. Não o foram porque também subscrevemos a máxima de que o texto estruturado é a forma mais clara, simples e eficiente de expor um programa de PLC, sobretudo quando há que fazê-lo para um público heterogéneo.

A norma IEC 61131-3 inclui a linguagem gráfica SFC (Sequential Function Chart) com o propósito de estruturar, sequenciar e desenvolver um programa de controlo. Consequentemente, o SFC não é uma alternativa ao texto estruturado ou a qualquer uma das outras linguagens previstas na norma. A linguagem SFC é um dos recursos mais interessantes e importantes da norma IEC 61131-3, sendo muitas vezes confundida com o *GRAFSET* tal como definido na primeira versão da norma IEC 60484 em virtude da semelhança semântica e gráfica destas. O facto do *GRAFSET* ser uma importante porta de acesso à compreensão de alguns aspectos da norma IEC 61131-3, nomeadamente em termos de organização de código, é mais uma razão para a sua ampla utilização neste texto em detrimento de outras ferramentas e metodologias de especificação.

Se o PLC que utiliza é minimamente compatível com a norma IEC 61131-3 mas só suporta as linguagens mais clássicas ou se, de qualquer modo, as pretende continuar a utilizar, então tem nestes exercícios um outro ponto de interesse: a tradução das soluções apresentadas para a “sua” linguagem de programação, mas usando a semântica e simbologia prevista na norma IEC 61131-3. Há, afinal, muitos motivos de interesse neste livro de exercícios!

Por último, se é adepto do texto estruturado e desenvolve habitualmente a sua programação de acordo com a norma IEC 61131-3, vai certamente considerar os

programas apresentados desleigantes, ineficientes e até algo desconformes com o espírito da norma. Concordando em absoluto que o texto estruturado e a norma IEC 61131-3 permitem soluções bem mais interessantes do que as apresentadas – por exemplo, usando “enumerated data types” para lidar com evoluções de máquinas de estados, ou encapsulando e reutilizando sistematicamente código do utilizador em funções e blocos de funções – há que dizer que tais abordagens lançariam, por certo, grandes confusões nos desconhecedores da norma e da linguagem de programação utilizada. E as soluções apresentadas pretendem ter dois méritos: primeiro, e mais fundamental, serem compreensíveis para qualquer programador de PLCs, mesmo que só minimamente habilitado; segundo, proporcionarem a cada leitor o exercício complementar de as traduzirem ou optimizarem para a(s) sua(s) linguagem(ns) de interesse. Consequentemente, se é fã do texto estruturado, encare as soluções apresentadas como pedaços de código a optimizar de acordo com a norma IEC 61131-3 e, já agora, também com a norma IEC 61499.

## Literatura e Materiais de Apoio

Caso pretenda iniciar-se na norma IEC 61131 e no “texto estruturado” tem múltiplos pontos de partida:

Encontrar a norma IEC 61131 numa biblioteca não deverá ser difícil. Outra hipótese é adquiri-la junto da “International Electrotechnical Commission” – <http://www.iec.ch>.

A parte mais interessante da norma é a Parte 3, toda ela dedicada às Linguagens de Programação. Tem a designação IEC 61131-3. É apresentada, discutida e exemplificada em diversos livros. Entre eles, destacam-se os seguintes:

- Robert W. Lewis – *“Programming Industrial Control Systems Using IEC 1131-3”* (IEE Control Engineering Series), 1998- ISBN: 0-85296-950-3;
- Karl-Heinz John – *“IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision-Making Tools”*. 1995, Springer Verlag. ISBN 3-540-67752-6. Parte deste livro está disponível na Internet a partir da página pessoal do autor: <http://www.fen-net.de/karlheinz.john/>.

Um sítio obrigatório em matéria de normas e PLCs é o da PLCOPEN – [www.plcopen.org](http://www.plcopen.org). Múltiplas informações úteis podem aqui ser encontradas.

Uma consulta ao Google, ou a qualquer outro motor de busca, sobre o tema “structured text programming” conduz a uma lista infindável de sítios e artigos interessantes. Entre eles, o livro *“Automating Manufacturing Systems with PLCs”* por Hugh Jack, disponível, à

data desta publicação, no sítio [claymore.engineer.gvsu.edu/~jackh/books/plcs](http://claymore.engineer.gvsu.edu/~jackh/books/plcs) e cuja leitura é absolutamente recomendada.

Se não possui um PLC compatível com a norma IEC 61131-3, mas gostaria de experimentar a programação em texto estruturado, então sugere-se uma viagem ao mundo dos “soft PLCs”. Muitos destes pacotes de software são totalmente compatíveis com a norma IEC 61131-3. Têm um custo elevado, mas os seus fabricantes disponibilizam versões gratuitas de demonstração, muito bem documentadas e exemplificadas, que, embora naturalmente limitadas, constituem ótimos pontos de partida para a programação segundo a norma IEC 61131-3. Sugere-se que conheça os seguintes produtos e fabricantes:

- ISaGRAF – <http://www.isagraf.com>;
- MULTIPROG – <http://www.kw-software.com>;
- CoDeSys – <http://www.3s-software.com>;
- TwinCAT PLC – <http://www.beckhoff.com>.

Se não conhece os “soft PLCs”, então este é um excelente pretexto para, finalmente, contactar com este interessantíssimo meio de controlo de muitas das modernas instalações industriais. Mais um ponto de interesse deste livro de exercícios!

## Acerca das Soluções

Tecidas algumas considerações sobre a norma IEC 61131-3 é, finalmente!, tempo de passar à divulgação das soluções dos problemas propostos.

Cada resolução compreende uma explicação e justificação, mais ou menos breve, dos procedimentos utilizados, seguida do respectivo programa em texto estruturado, devidamente comentado. Porque, na generalidade dos casos, as tarefas se vão complementando à medida que uma missão avança, a solução de uma tarefa é vulgarmente o ponto de partida para a seguinte. Consequentemente, grande parte do código desenvolvido numa tarefa tende a ser reutilizado nas seguintes. Daí, volta-se a frisar, a importância de compreender bem a solução de uma tarefa antes de avançar para a seguinte.

No caso de soluções menos triviais ou propensas a alguma ambiguidade, os programas são desenvolvidos com base numa especificação prévia em *GRAFSET*, devidamente comentada. É importante referir que tais especificações seguem a segunda edição da norma IEC 60484. Dado que esta versão é relativamente recente, é provável que alguns leitores não conheçam ainda suficientemente bem as alterações introduzidas nesta

versão, podendo daí resultar alguma estranheza relativamente a alguns aspectos gráficos e funcionais de uns tantos *grafcets*; por exemplo, a afectação de acções a algumas transições. Embora, e a pensar exactamente nos leitores menos familiarizados com a versão em vigor, as soluções apresentadas apontem e alertem para algumas das inovações introduzidas na segunda edição da norma IEC 60484, tais esclarecimentos não dispensam a consulta da mesma. A promoção da segunda edição da norma IEC 60484 que, relativamente à primeira, se distanciou bastante da linguagem SFC, é outro propósito deste livro!

A declaração e inicialização das variáveis é um aspecto já de si importante em qualquer programa, mas é uma questão particularmente relevante quando se programa de acordo com a norma IEC 61131-3. Nesse contexto, a declaração das variáveis merece um espaço próprio nas soluções apresentadas, tal como deve também merecer uma atenção especial por parte do leitor. E isto porque, a declaração das variáveis compreende também a inicialização, implícita ou explícita, das mesmas com um valor de interesse para os programas que as utilizam. Como tal, a declaração de uma variável não deve ser vista como um mero formalismo de pouco interesse para as soluções apresentadas, mas antes como uma informação importante. Por exemplo, a atribuição do valor inicial TRUE a uma variável de estado revela, ou confirma, que a mesma simboliza uma etapa ou estado inicial.

Por questão de organização, as variáveis estão divididas em dois grupos: as que são comuns a todas as missões e as que apenas são válidas dentro cada missão. O primeiro grupo é composto exclusivamente pelas variáveis de I/O. São apresentadas no ponto seguinte. Mais à frente, já na esfera das soluções de cada missão, surge a declaração das variáveis e das instâncias de blocos funcionais utilizadas nas tarefas afectas à missão em causa.

Como nota final, importa sublinhar, mais uma vez, que as soluções apresentadas são apenas “soluções possíveis” e, de forma alguma, as únicas ou as melhores. Sugere-se, por isso, que o leitor as compare com as suas, tentando listar vantagens e inconvenientes relativos. A elaboração de um caderno de notas reflectindo as conclusões pessoais sobre a resolução de cada tarefa é a última proposta deste livro de exercícios e, sem dúvida, uma das mais importantes para formadores e formandos.

## **Declaração das Variáveis de I/O**

As variáveis que representam os parâmetros de entrada e de saída de um programa são globalmente designadas por variáveis de I/O. As variáveis de I/O são fundamentais em qualquer programa, já que são o meio de troca de informação entre este e os seus

elementos periféricos, físicos ou lógicos, tais como sensores, actuadores, sinalizadores ou outras aplicações de software.

No caso da aplicação “ITS PLC”, a correcta troca de informação entre os sistemas virtuais e o PLC externo é uma condição necessária ao seu correcto funcionamento. Para tal, é necessário que ao PLC cheguem informações sobre o estado da instalação, na forma de variáveis de entrada do PLC, e que o estado da instalação se altere de acordo com acções de controlo adequadas, reflectidas nas variáveis de saída do PLC. Esta troca de informação requer uma ligação física entre o PLC e a placa de interface USB que acompanha o produto. É dessa ligação física que resulta o mapeamento das variáveis de I/O do PLC nas variáveis de I/O dos sistemas virtuais “ITS PLC”.

A declaração das variáveis de I/O tem três propósitos: atribui-lhes nomes que podem ser usados na escrita do programa, definir os tipos de variáveis em jogo e afectá-las aos endereços físicos de I/O do PLC. Consequentemente, ao declarar as variáveis, cada programador tenderá a apelidá-las a seu gosto e a mapeá-las em função do PLC utilizado. De comum a todas as declarações, haverá apenas a afirmação de que cada ponto de I/O corresponde a uma variável booleana.

A norma IEC 61131-3 prevê a hipótese de, na declaração de uma variável, afectá-la a um espaço físico do PLC (memória, entrada ou saída). Esta hipótese, particularmente interessante na declaração das variáveis de I/O reflecte-se no emprego do atributo “AT”. Assim, assumindo um PLC com módulos de entrada e saída de 8 “bits”, a declaração das variáveis de I/O pode ser feita do seguinte modo:

```
(*****  
    Declaração das Variáveis de Entrada e Saída  
*****)
```

```
VAR_INPUT
```

```
In_0    AT %IX0.0 : BOOL; (* Sensor 0 *)  
In_1    AT %IX0.1 : BOOL; (* Sensor 1 *)  
In_2    AT %IX0.2 : BOOL; (* Sensor 2 *)  
In_3    AT %IX0.3 : BOOL; (* Sensor 3 *)  
In_4    AT %IX0.4 : BOOL; (* Sensor 4 *)  
In_5    AT %IX0.5 : BOOL; (* Sensor 5 *)  
In_6    AT %IX0.6 : BOOL; (* Sensor 6 *)  
In_7    AT %IX0.7 : BOOL; (* Sensor 7 *)  
In_8    AT %IX1.0 : BOOL; (* Sensor 8 *)  
In_9    AT %IX1.1 : BOOL; (* Sensor 9 *)  
In_10   AT %IX1.2 : BOOL; (* Sensor 10 *)  
In_11   AT %IX1.3 : BOOL; (* Selector de Modo Manual/Automático *)  
In_12   AT %IX1.4 : BOOL; (* Botoneira Iniciar *)  
In_13   AT %IX1.5 : BOOL; (* Botoneira Parar *)  
In_14   AT %IX1.6 : BOOL; (* Botoneira Reiniciar *)  
In_15   AT %IX1.7 : BOOL; (* Botoneira Emergência *)
```

END\_VAR

VAR\_OUTPUT

```
Out_0    AT %QX0.0 : BOOL; (* Actuador 0 *)
Out_1    AT %QX0.1 : BOOL; (* Actuador 1 *)
Out_2    AT %QX0.2 : BOOL; (* Actuador 2 *)
Out_3    AT %QX0.3 : BOOL; (* Actuador 3 *)
Out_4    AT %QX0.4 : BOOL; (* Actuador 4 *)
Out_5    AT %QX0.5 : BOOL; (* Actuador 5 *)
Out_6    AT %QX0.6 : BOOL; (* Actuador 6 *)
Out_7    AT %QX0.7 : BOOL; (* Actuador 7 *)
Out_8    AT %QX1.0 : BOOL; (* Luz da botoneira Iniciar *)
Out_9    AT %QX1.1 : BOOL; (* Luz da botoneira Reiniciar *)
```

END\_VAR

```
(*****
                                     Fim da declaração
*****)
```

A generalidade dos ambientes de programação de PLCs prevê procedimentos de declaração de variáveis mais ou menos semelhantes ao representado. Mas, admitindo que a declaração pode ser confusa para os leitores menos familiarizados com a norma IEC 61131-3, o que importa reter desta declaração é o seguinte:

- Em todas as tarefas, de todas as missões, as variáveis de entrada booleanas In\_0, In\_1, ..., In\_10 mapeiam, por esta ordem, os valores lógicos produzidos pelos sensores virtuais 0, 1, ..., 10;
- Em todas as tarefas, de todas as missões, as variáveis de entrada booleanas In\_11, In\_12, ..., In\_15 mapeiam, por esta ordem, o estado do selector de modo e das botoneiras Iniciar, Parar, Reiniciar e Emergência;
- Em todas as tarefas, de todas as missões, as variáveis de saída booleanas Out\_0, Out\_1, ..., Out\_7 mapeiam, por esta ordem, os valores lógicos dos actuadores virtuais 0, 1, ..., 7;
- Em todas as tarefas, de todas as missões, as variáveis de saída booleanas Out\_8 e Out\_9 mapeiam os valores lógicos dos sinalizadores luminosos das botoneiras Iniciar e Reiniciar, respectivamente.

A norma IEC 61131-3 possibilita a inicialização de uma variável aquando da sua declaração. Esta possibilidade não tem interesse para as variáveis de I/O e, nem sequer é possível, porque não faz sentido, para as variáveis de entrada. Por esse motivo, as variáveis de I/O não são inicializadas na respectiva declaração.



---

# Missão 1: Automatização de uma estação de transporte e triagem de mercadorias em paletes

---

## Variáveis e instâncias de blocos funcionais usadas na Missão 1

A cada tarefa de cada missão corresponde um programa que emprega um conjunto de variáveis e instâncias de blocos de funções, cuja declaração é exigida pela norma IEC 61131-3. Porém, apresentar as soluções das diferentes tarefas seguindo este princípio, levaria a que o código de cada programa fosse antecedido de uma lista de variáveis e instâncias de blocos funcionais, por vezes extensa, com muitas delas já introduzidas e explicadas nas soluções de tarefas anteriores.

Para evitar sucessivas e fastidiosas declarações, que em pouco ou nada contribuiriam para a compreensão dos sucessivos programas, e assim aligeirar o documento, optou-se por reunir, numa única declaração, as variáveis e as instâncias de blocos funcionais relativas aos doze programas de cada missão. Essa declaração conjunta, que abre a apresentação das soluções de cada missão, é feita em conformidade com a norma IEC 61131-3, e deve ser tomada em devida conta na apreciação do código correspondente à solução de cada tarefa. Em particular, é de especial importância a observação do valor inicial atribuído a cada variável.

De acordo com a norma IEC 61131-3, a definição do valor inicial de uma variável permite atribuir-lhe um valor diferente daquele que seria o seu valor inicial por defeito, o qual é função do tipo de variável; por exemplo, variáveis booleanas, inteiras e reais iniciam-se, por defeito, em 0. Parecerá, pois, estranho que, na declaração de algumas variáveis, lhes sejam atribuídos os valores que teriam por defeito; por exemplo, é possível observar que muitas variáveis booleanas são inicializadas em FALSE. Tal “redundância” tem o propósito de revelar explicitamente o valor inicial de algumas variável particularmente

importantes. Por exemplo, no caso de uma variável de estado, o seu valor inicial deve tipicamente reflectir uma condição inicial prevista no cenário em que a tarefa se desenrola; logo, tem de ser inicializada com TRUE ou FALSE, em conformidade. Se essa inicialização for bem explícita, tanto melhor...

Assim, e muito concretamente, optou-se por inicializar explicitamente mesmo aquelas variáveis que implicitamente seriam inicializadas com os valores de interesse porque, mostra a experiência, quando a inicialização de uma variável não é explícita, alguns leitores têm tendência a deduzir que o valor inicial dessa variável é indiferente, não tomando desse modo consciência da importância do valor inicial que, por defeito, lhe é atribuído. Em coerência com este princípio, entendeu-se inicializar explicitamente todas as variáveis aquando da sua declaração, excepto aquelas cujo valor inicial é, de facto, irrelevante.

Os casos em que a mesma variável deve ser inicializada com valores distintos em diferentes tarefas são devidamente salientados nos comentários que acompanham a respectiva declaração. Também são salientados os casos em que os valores iniciais são apenas exemplificativos; por exemplo, parâmetros de configuração, tabelas de receitas, etc. Em qualquer caso, a declaração de uma variável encerra em comentário o significado da mesma, a codificação emprega e as tarefas em que é utilizada. As variáveis são agrupadas em função do seu contexto e, dentro de cada grupo, listadas por ordem alfabética. Tal permite enquadrar devidamente a funcionalidade das diferentes variáveis e encontrar com facilidade a declaração de uma qualquer variável. Por último, há que referir que a declaração das variáveis antecede a declaração das instâncias dos blocos funcionais.

Feitas estas explicações, segue-se a declaração das variáveis empregues nas doze tarefas que compõem a Missão 1, a qual corresponde à seguinte listagem:

```
(*****  
    Declaração das variáveis e instâncias de  
    blocos funcionais IEC 61131-3 usadas na Missão 1  
*****)  
  
VAR    (* Declaração e inicialização de variáveis *)  
  
(* Variáveis afectas ao comando do tapete de entrada *)  
  
Busy_1 : BOOL := FALSE;          (* Tarefas 2-3, 5-12 *)  
    (* Disponibilidade do tapete de entrada: 0 -> Disponível *)  
Count  : WORD := 16#8000;        (* Tarefas 3, 5-12 *)  
    (* Contador em anel, inicialmente a zero -> 8000H *)  
Fila   : WORD;                  (* Tarefas 8-12 *)  
    (* Fila das alturas das paletes no tapete de entrada *)  
Mem_1  : BOOL := FALSE;         (* Tarefas 1-3, 5-12 *)  
    (* Estado do tapete de entrada: 0 -> Sem paletes *)
```

```

Mem_timer : BOOL;                                (* Tarefas 9-12 *)
    (* Memória para retenção de Timer_1 *)

(* Variáveis afectas ao comando da mesa rotativa *)

Busy_2 : BOOL := FALSE;                          (* Tarefas 5-12 *)
    (* Disponibilidade da mesa rotativa: 0 -> Disponível *)
Carrega : BOOL := FALSE;                        (* Tarefas 4-12 *)
    (* Estado possível da mesa rotativa *)
Descarrega : BOOL := FALSE;                     (* Tarefas 4-12 *)
    (* Estado possível da mesa rotativa *)
Livre : BOOL := TRUE;                           (* Tarefas 4-12 *)
    (* Estado inicial da mesa rotativa *)
Mem_timeout_mesa : BOOL := FALSE;                (* Tarefas 11-12 *)
    (* Memória para retenção de Timeout_mesa *)
Roda_carregada : BOOL := FALSE;                 (* Tarefas 4-12 *)
    (* Estado possível da mesa rotativa *)
Roda_descarregada : BOOL := FALSE;              (* Tarefas 4-12 *)
    (* Estado possível da mesa rotativa *)
Sentido_descarga : BOOL := FALSE;               (* Tarefas 6-12 *)
    (* Se 0, a descarga é para tapete da direita *)

(* Variáveis afectas ao comando do tapete de saída da direita *)

Busy_6 : BOOL := FALSE;                          (* Tarefas 7-12 *)
    (* Disponibilidade do tapete de saída da direita:
    0 -> Disponível *)
Mem_6 : BOOL := FALSE;                           (* Tarefas 5-12 *)
    (* Estado do tapete de saída da direita: 0 -> Sem paletes *)
Mem_timeout_direita : BOOL := FALSE;            (* Tarefas 11-12 *)
    (* Memória para retenção de Timeout_direita *)

(* Variáveis afectas ao comando do tapete de saída da esquerda *)

Busy_5 : BOOL := FALSE;                          (* Tarefas 7-12 *)
    (* Disponibilidade do tapete de saída da esquerda:
    0 -> Disponível *)
Mem_5 : BOOL := FALSE;                           (* Tarefas 6-12 *)
    (* Estado do tapete de saída da esquerda: 0 -> Sem paletes *)
Mem_timeout_esquerda : BOOL := FALSE;           (* Tarefas 11-12 *)
    (* Memória para retenção de Timeout_esquerda *)

(* Variáveis afectas aos modos de funcionamento da instalação *)

Automatico : BOOL := FALSE;                      (* Tarefas 10-12 *)
    (* Estado possível da instalação - inicial na Tarefa 10 *)
Encerramento : BOOL := FALSE;                  (* Tarefas 10-12 *)
    (* Estado possível da instalação *)
Limpeza : BOOL := FALSE;                        (* Tarefas 11-12 *)
    (* Estado possível da instalação *)
Parada : BOOL;                                  (* Tarefas 10-12 *)
    (* Instalação parada *)
Pronto : BOOL := FALSE;                         (* Tarefas 10-12 *)

```

```

(* Estado possível da instalação *)
Standby : BOOL := TRUE; (* Tarefas 11-12 *)
(* Estado possível da instalação - inicial nas Tarefas 11-12 *)

(* Instâncias de Blocos de Funções *)

(* Detecção de transições descendentes *)

F_In_0 : F_TRIG; (* Tarefas 2-3, 5-12 *)
(* Transição descendente de In_0 *)
F_In_3 : F_TRIG; (* Tarefas 1-12 *)
(* Transição descendente de In_3 *)
F_In_7 : F_TRIG; (* Tarefas 4-12 *)
(* Transição descendente de In_7 *)
F_In_8 : F_TRIG; (* Tarefas 6-12 *)
(* Transição descendente de In_8 *)
F_In_9 : F_TRIG; (* Tarefas 5-12 *)
(* Transição descendente de In_9 *)
F_In_10 : F_TRIG; (* Tarefas 6-12 *)
(* Transição descendente de In_10 *)

(* Detecção de transições ascendentes *)

R_In_0 : R_TRIG; (* Tarefa 1 *)
(* Transição ascendente de In_0 *)
R_Limpeza : R_TRIG; (* Tarefas 11-12 *)
(* Transição ascendente de Limpeza *)
R_Parada : R_TRIG; (* Tarefas 11-12 *)
(* Transição ascendente de Parada - Instalação parou *)

(* Temporizadores *)

Timer_1 : TON; (* Tarefas 9-12 *)
(* Usado no comando do tapete de entrada *)
Pisca_encerra_auto_1 : TON; (* Tarefa 12 *)
(* Temporizador 1 do Pisca_encerra_auto *)
Pisca_encerra_auto_2 : TON; (* Tarefa 12 *)
(* Temporizador 2 do Pisca_encerra_auto *)
Pisca_encerramento_1 : TON; (* Tarefas 10-12 *)
(* Temporizador 1 do Pisca_encerramento *)
Pisca_encerramento_2 : TON; (* Tarefas 10-12 *)
(* Temporizador 2 do Pisca_encerramento *)
Pisca_limpeza_1 : TON; (* Tarefas 11-12 *)
(* Temporizador 1 do Pisca_limpeza *)
Pisca_limpeza_2 : TON; (* Tarefas 11-12 *)
(* Temporizador 2 do Pisca_limpeza *)
Timeout_direita : TON; (* Tarefas 11-12 *)
(* Timeout no comando do tapete de saída da direita *)
Timeout_entrada : TON; (* Tarefas 11-12 *)
(* Timeout no comando do tapete de entrada *)
Timeout_esquerda : TON; (* Tarefas 11-12 *)
(* Timeout no comando do tapete de saída da esquerda *)
Timeout_mesa : TON; (* Tarefas 11-12 *)

```

```
(* Timeout no comando da mesa rotativa *)

(* Contadores *)

Conta_paletes : CTD; (* Tarefa 12 *)
  (* Contador de paletes *)

END_VAR

(*****
  Fim da declaração
  *****)
```

## Resolução da Tarefa 1

De acordo com o enunciado, o tapete de entrada deve movimentar-se quando nele existe uma paleta. Então, admitindo que é possível representar o estado do tapete por uma variável binária (ou memória), “Mem\_1”, tal que Mem\_1 = 1 se existe uma paleta no tapete, o comando deste transportador passa simplesmente pela expressão Out\_1 = Mem\_1.

Dado que não existe um sensor que indique se o tapete tem ou não uma paleta, o valor lógico de Mem\_1 terá de ser deduzido; concretamente, com base na observação da entrada e saída de paletes no tapete. Assim, a entrada de uma paleta deve fazer Mem\_1 = 1 – ou seja, o “set” de Mem\_1; a saída de uma paleta deve fazer Mem\_1 = 0 – ou seja, o “reset” de Mem\_1. Mem\_1 deve ser inicializada em 0 uma vez que é dito que inicialmente não há paletes no tapete de entrada.

Em termos lógicos, a entrada e saída de paletes no tapete reflectem-se na transição ascendente de In\_0 ( $\uparrow$ In\_0 = 1) e na transição descendente de In\_3 ( $\downarrow$ In\_3 = 1), respectivamente. O “set” e o “reset” de Mem\_1 devem pois ser motivados por estas transições.

O programa correspondente à presente tarefa é o seguinte:

```
(*****  
          Missão 1 - Tarefa 1  
*****)  
  
(**** DETECÇÃO DE EVENTOS RELEVANTES ****)  
  
R_In_0 (CLK := In_0);  
(* R_In_0.Q é 1 quando In_0 tem uma transição ascendente. Sinaliza que  
uma paleta chegou ao tapete de entrada *)  
  
F_In_3 (CLK := In_3);  
(* F_In_3.Q é 1 quando In_3 tem uma transição descendente. Sinaliza que  
uma paleta abandonou o tapete de entrada *)  
  
(**** COMANDO DO TAPETE DE ENTRADA ****)  
  
(* Definição da variável de estado, Mem_1, em função dos sinais de  
entrada *)  
  
IF R_In_0.Q THEN          (* Quando chega uma paleta *)  
    Mem_1 := TRUE;        (* "Set" de Mem_1 *)  
END_IF;  
  
IF F_In_3.Q THEN          (* Quando sai uma paleta *)  
    Mem_1 := FALSE;       (* "Reset" de Mem_1 *)  
END_IF;
```

```
(* Definição da variável de saída, Out_1, em função do estado do tapete  
de entrada *)
```

```
Out_1 := Mem_1;
```

```
(*****
```

```
        Fim do Programa
```

```
*****)
```

## Resolução da Tarefa 2

Este problema é bastante simples, mas encerra ideias de importância fundamental para a generalidade das tarefas da presente missão – na realidade, para a generalidade dos problemas de transporte de mercadorias em passareiras.

É muito importante notar que o transporte de uma paleta tem três fases. Veja-se para o caso do tapete de entrada:

- A carga da paleta – requer o movimento simultâneo do alimentador e do tapete de entrada. Inicia-se quando a paleta é detectada pelo sensor 0 e termina quando o mesmo sensor deixa de a detectar. Assim, durante esta fase, o valor lógico de  $In_0$  é 1;
- O transporte, propriamente dito – requer unicamente o movimento do tapete de entrada. Durante esta fase, a paleta não é detectada pelos sensores colocados nos extremos do tapete, pelo que  $In_0 = In_3 = 0$ ;
- A descarga da paleta – requer o movimento simultâneo do tapete de entrada e dos rolos da mesa rotativa. Inicia-se quando a paleta é detectada pelo sensor 3 e termina quando o mesmo sensor deixa de a detectar. Assim, durante esta fase, o valor lógico de  $In_3$  é 1.

Porque a carga do tapete de entrada corresponde à descarga do alimentador – tal como a descarga do tapete de entrada corresponde à carga da mesa rotativa – então, estando o tapete de entrada indisponível, isto é, transportando ele uma paleta, o alimentador não pode descarregar. Ou seja, o alimentador não pode passar uma paleta para além da posição do sensor 0. Só o poderá fazer quando o tapete de entrada ficar disponível, isto é, quando nele não existir qualquer paleta.

Uma solução genérica, para problemas deste tipo, é aceitar que um tapete A gera o sinal  $Busy_A = 1$  quando, por qualquer razão, está indisponível para receber paletes. Ao ler este sinal, um tapete B, que alimente A, fica a saber que não deve movimentar paletes para além do seu limite. Só o poderá fazer quando o tapete A produzir o sinal  $Busy_A = 0$ .

A adaptação deste princípio ao problema em causa é muito simples: o controlador do tapete de entrada faz o “set” de  $Busy_1$  após carregar uma paleta – isto é, quando sente uma transição descendente de  $In_0$  – e o “reset” de  $Busy_1$  sempre que uma paleta é descarregada – isto é, quando sente a transição descendente de  $In_3$ . Compete ao comando do alimentador ler o sinal  $Busy_1$  de modo que, chegando uma paleta ao fim do alimentador e estando o tapete de entrada ocupado, o movimento seja suspenso até que o tapete de entrada fique disponível.

Outro aspecto importante é que, por questões de eficiência, o tapete de entrada não deve parar após descarregar uma paleta caso, nessa altura, já exista outra paleta no final do alimentador. Assim, as condições de “set”/“reset” de Mem\_1 têm de ser modificadas relativamente à Tarefa 1. Uma hipótese é só fazer o “reset” na transição descendente de In\_3 se, nesse instante, In\_0 = 0; outra, é fazer o “set” quando In\_0 = 1 e torná-lo dominante sobre o “reset”. A solução apresentada segue a segunda hipótese.

```

(*****
      Missão 1 - Tarefa 2
*****)

(**** DETECÇÃO DE EVENTOS RELEVANTES ****)

F_In_0(CLK := In_0);
      (* Paleta foi carregada no tapete de entrada
      - ou seja, foi descarregada do alimentador *)
F_In_3(CLK := In_3);

(**** COMANDO DO TAPETE DE ENTRADA ****)

(* Definição de Mem_1 e de Busy_1 *)

IF F_In_3.Q THEN      (* Quando uma paleta abandona o tapete *)
    Mem_1 := FALSE;   (* "Reset" de Mem_1 *)
    Busy_1 := FALSE;  (* "Reset" de Busy_1 *)
END_IF;

IF F_In_0.Q THEN      (* Quando uma paleta é carregada *)
    Busy_1 := TRUE;   (* "Set" de Busy_1 *)
END_IF;

IF In_0 THEN          (* Havendo paleta no fim do alimentador *)
    Mem_1 := TRUE;    (* "Set" de Mem_1 *)
END_IF;

(** NOTA IMPORTANTE:
Note que o set de Mem_1 é agora para In_0 = 1 e dominante sobre o
reset - uma vez que a instrução de set surge depois da de reset. Então,
se F_In_3.Q e In_0 forem ambos verdadeiros, Mem_1 fica com o valor
lógico TRUE que entregará à saída Out_1. Isso garante que o tapete de
entrada continua em movimento, sem transições a zero, caso já exista uma
paleta à saída do alimentador quando a anterior abandona o tapete de
entrada **)

(* Definição da saída Out_1 - como na Tarefa 1 *)
Out_1 := Mem_1;

(**** COMANDO DO ALIMENTADOR ****)

(* Alimentador não está em movimento se tapete de entrada ocupado e há
uma paleta no final do alimentador *)
Out_0 := NOT Busy_1 OR NOT In_0;

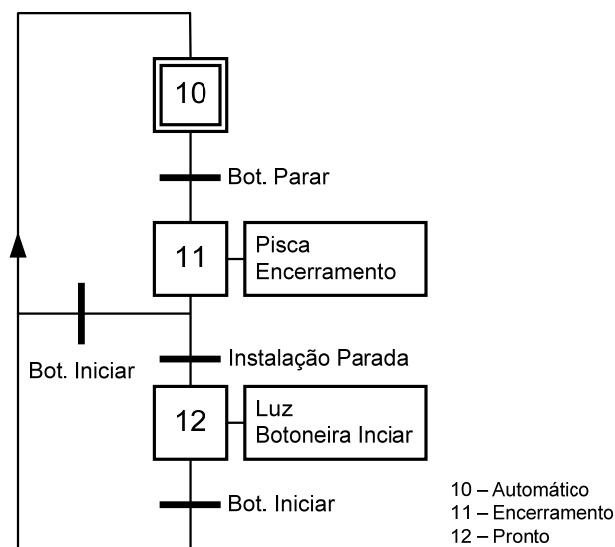
```

```
(* O mesmo que: Out_0 := NOT (Busy_1 AND In_0) *)  
  
(*****  
      Fim do Programa  
*****)
```

## Resolução da Tarefa 10

A instalação passou a ter três estados (ou fases) possíveis, cuja evolução está representada no *grafcet* da Figura M1T10a:

- Automático – Funcionamento contínuo como na Tarefa 9;
- Encerramento – Alimentador não fornece paletes e os demais transportadores transferem as paletes ainda em trânsito;
- Pronto – A instalação não funciona por escassez de paletes.



**Figura M1T10a – Evolução e sinalização dos modos de funcionamento da instalação**

O curioso deste funcionamento é que, exceptuando o alimentador, todos os transportadores, em todos dos modos, se comportam da forma descrita na Tarefa 9! É por simples “escassez de paletes” – ou seja, pelo facto do alimentador deixar de fornecer paletes – que a instalação entra na fase de encerramento, acabando depois inactiva. Fazer a instalação regressar ao modo automático, é permitir que o alimentador volte a fornecer paletes que os diferentes transportadores encaminham devidamente.

Assim, o código do comando do alimentador – que permaneceu imutável desde a Tarefa 2 – é o único que requer alteração em relação à tarefa anterior. Considerando que, caso exista uma paleta na zona do sensor 0 no momento em que é dada ordem de encerramento, essa paleta é ainda enviada para o tapete de entrada, o comando do alimentador passa a ser dado por:

```
Out_0 = (NOT In_0 OR NOT Busy_1) AND (In_0 OR NOT Encerramento) AND NOT Pronto;
```

Ou, equivalentemente:

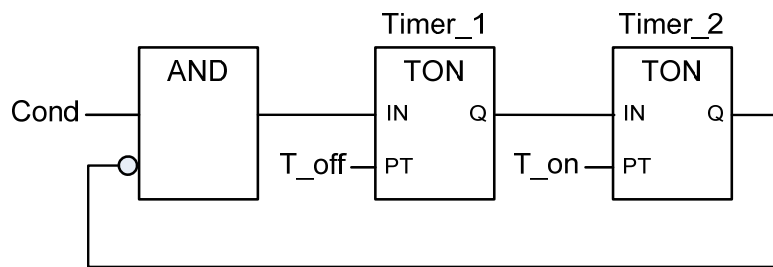
```
Out_0 = (NOT In_0 OR NOT Busy_1) AND (In_0 OR Automatico);
```

As variáveis “Automático”, “Encerramento” e “Pronto” são variáveis internas que tomam o valor lógico 1 quando a instalação está no respectivo modo de funcionamento, permitindo programar facilmente o *grafcet* da Figura M1T10a. Reconhecer que a instalação está parada é reconhecer que nenhum transportador contém paletes:

```
Parada = NOT Mem_1 AND Livre AND NOT Mem_5 AND NOT Mem_6;
```

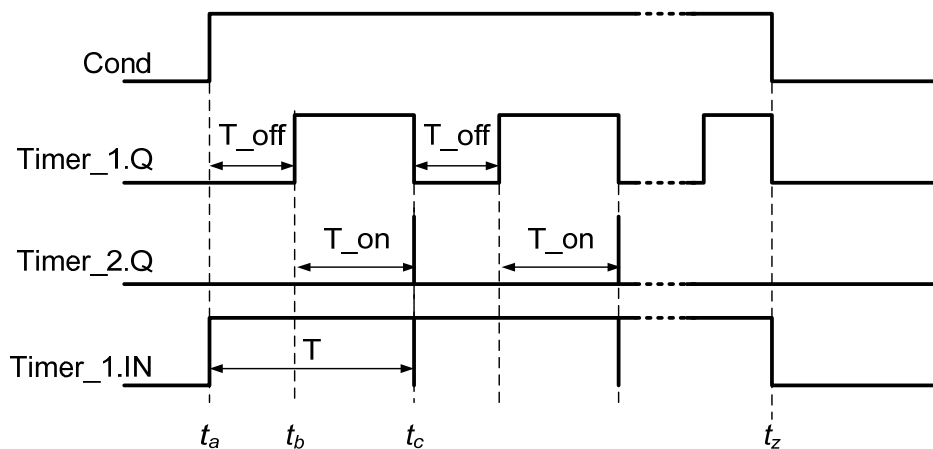
Para sinalizar o encerramento da instalação é necessário realizar um pisca com as características temporais desejadas que só funcione quando a variável Encerramento exibe o valor lógico 1.

A forma mais simples de realizar um pisca é através de dois temporizadores do tipo “on-delay” com a configuração mostrada na Figura M1T10b:



**Figura M1T10b – Pisca construído com dois temporizadores “on-delay”**

A compressão deste esquema passa pela compreensão do diagrama temporal representado na Figura M1T10c:



**Figura M1T10c – Diagrama temporal para o pisca da Figura M1T10b**

Considere-se que, inicialmente, a condição de interesse genérica, Cond, que deverá fazer funcionar o pisca, está a 0. Isso faz com que as entradas e saídas dos temporizadores estejam também em 0.

O primeiro temporizador, Timer\_1, é armado ao tempo  $t_a$ , ou seja, quando a condição Cond – no caso desta tarefa será a variável “Encerramento” – toma o valor lógico 1. Mantendo-se Cond a 1, a saída de Timer\_1 vai a 1 ao tempo  $t_b = t_a + T_{off}$ , fazendo Timer\_2 iniciar a respectiva temporização. Consequentemente, mantendo-se Cond a 1, a saída do segundo temporizador vai a 1 ao tempo  $t_c = t_b + T_{on}$ , ou seja, um tempo  $T_{on}$  após o primeiro, desarmando-o. O desarme do primeiro temporizador desarma o segundo que, por sua vez, rearma o primeiro. Significa isto que, ao tempo  $t_c$  retomam-se as condições observadas em  $t_a$ , pelo que o ciclo se repetirá com uma periodicidade  $T = T_{on} + T_{off}$ .

Assim, a saída do primeiro temporizador alternará com um período  $T = T_{off} + T_{on}$  e um “duty cycle” =  $T_{on}/T$  até ao tempo  $t_z$ , instante em que a variável de interesse, Cond, passa a exibir o valor lógico 0.

Assumindo-se que, no caso da presente tarefa, o pisca será realizado por dois temporizadores designados por Pisca\_encerramento\_1 e Pisca\_encerramento\_2, então a sinalização do estado da instalação através da luz da botoneira Iniciar é feita pela seguinte instrução:

```
Out_8 = Parada OR Pisca_encerramento_1.Q;
```

O programa correspondente à Tarefa 10 é o seguinte:

```
(*****
      Missão 1 - Tarefa 10
*****)

(**** DETECÇÃO DE EVENTOS RELEVANTES ****)

F_In_0(CLK := In_0);
F_In_3(CLK := In_3);
F_In_7(CLK := In_7);
F_In_8(CLK := In_8);
F_In_9(CLK := In_9);
F_In_10(CLK := In_10);

(**** EVOLUÇÃO E SINALIZAÇÃO DOS MODOS DE FUNCIONAMENTO ****)
(* Evolução das varáveis de estado *)

IF Automatico AND NOT In_13 THEN      (* Automático -> Encerramento *)
    Encerramento := TRUE;
    Automatico := FALSE;
END_IF;
```

```

IF Encerramento AND Parada THEN      (* Encerramento -> Pronto *)
    Pronto := TRUE;
    Encerramento := FALSE;
END_IF;

IF In_12 THEN      (* Quando a Botoneira Iniciar é premida *)
    Automatico := TRUE; (* Encerramento ou Pronto -> Automático *)
    Encerramento := FALSE;
    Pronto := FALSE;
END_IF;

(* Detecção de instalação inactiva *)
Parada := NOT Mem_1 AND Livre AND NOT Mem_5 AND NOT Mem_6;

(* Materialização do Pisca de Encerramento *)
Pisca_encerramento_1(IN := Encerramento AND NOT Pisca_encerramento_2.Q,
PT := T#1s);
Pisca_encerramento_2(IN := Pisca_encerramento_1.Q, PT := T#1s);

(* Sinalização do Modo *)
Out_8 := Pronto OR Pisca_encerramento_1.Q;

(**** COMANDO DO TAPETE DE SAÍDA DA ESQUERDA ****)
(* Como na Tarefa 7 *)

IF F_In_10.Q THEN
    Mem_5 := FALSE;
    Busy_5 := FALSE;
END_IF;

IF F_In_8.Q THEN
    Busy_5 := TRUE;
END_IF;

IF In_8 THEN
    Mem_5 := TRUE;
END_IF;

Out_5 := Mem_5 AND (NOT In_10 OR In_15);

(**** COMANDO DO TAPETE DE SAÍDA DA DIREITA ****)
(* Como na Tarefa 7 *)

IF F_In_9.Q THEN
    Mem_6 := FALSE;
    Busy_6 := FALSE;
END_IF;

IF F_In_7.Q THEN
    Busy_6 := TRUE;
END_IF;

```