Table of contents

Information for readers 13

1 Developments in the Field of Control Engineering 18
1.1 The early days of programmable logic controllers (PLCs) 19
1.2 The PLC learns to communicate 22
1.3 Development of fieldbus systems 24
1.4 Integration of display systems in PLCs 25
1.5 Integration of motion control in PLCs 27
1.6 Drives become fully-fledged bus system nodes 30
1.7 PLC and PAC – what is the difference? 31
1.8 General conclusions about past developments 31

2 Basic Principles of Object-Oriented Programming 33
2.1 The basis of object-oriented programming 33
2.1.1 History 33
2.1.2 What's different? 34
2.1.3 What does object orientation mean? 35
2.1.4 Objects and their interactions 36
2.2 General principles of OOP 37
2.2.1 Objects 37
2.2.2 Classes 39
2.2.3 Inheritance 39
2.2.4 Overriding 41
2.2.5 Interfaces for object interaction 42
2.2.6 Summary 44
2.2.7 Advantages of using OOP 45
2.2.8 Disadvantages of OOP 45
2.3 Tips about defining classes 46

3 Object-Oriented Programming 49
3.1 Implementation of OOP with SIMOTION 49
3.2 Function blocks with methods 50
3.2.1 Modularization without OOP extensions 51
3.2.2 Program and data are separate 53
3.2.3 Advances in the life cycle of software 55
3.2.4 Disadvantages of programming without OOP extensions 56
3.2.5 Extensions to FBs and their access specification 57
3.2.6 Use of methods to improve program structuring 59
3.2.6.1 Example of FB with methods 60

3.2.6.2 Example of a function block call ... 61
3.2.7 Function block with methods for placing commands ... 62
3.2.7.1 Example of the FB with command methods ... 63
3.2.7.2 Example of an FB call with command methods ... 65
3.3 Classes (CLASS) ... 66
3.3.1 Keywords supported for a class ... 67
3.3.1.1 Example of a CLASS declaration ... 69
3.3.2 Methods (METHOD) ... 69
3.3.3 Methods and their access specification ... 70
3.3.4 Declaration of instances of a class ... 71
3.3.5 Rules for identifiers in a class ... 72
3.3.6 Use of class methods ... 72
3.3.6.1 Example of a CLASS COUNTER ... 73
3.3.6.2 Use of the method of CLASS COUNTER ... 74
3.3.6.3 Extension of the CLASS COUNTER and use of THIS ... 75
3.3.6.4 Use of the methods UP and DOWN ... 76
3.3.7 Classes and inheritance ... 76
3.3.7.1 Example of derivation of a class ... 78
3.3.7.2 Example of how to use base and derived classes ... 79
3.3.7.3 Other aspects of the method call ... 80
3.3.7.4 Example of base and derived classes in a function ... 81
3.3.8 Abstract classes ... 82
3.4 Examples of valve applications with OOP ... 84
3.4.1 Example with 4/3-way valve ... 84
3.4.1.1 Example of a class for 4/3-way valves ... 85
3.4.1.2 Example of a valve call ... 87
3.4.1.3 Example with 4/3-way valve with fast/slow speed ... 88
3.4.1.4 Example of a derived class ValveControl43FS ... 89
3.4.1.5 Example of calls of base class and extended class ... 90
3.4.1.6 Example of call of extended class with basic function ... 91
3.5 Interfaces ... 92
3.5.1 Supported features ... 93
3.5.2 Principles of interfaces ... 94
3.5.2.1 Example of an interface declaration ... 95
3.5.3 Representation of interfaces in the PNV of SCOUT ... 97
3.5.4 Benefits of interfaces ... 99
3.5.5 Interfaces as a reference to classes ... 100
3.5.6 Valve classes with interfaces ... 103
3.5.7 Declaration of the valve interface ... 105
3.5.7.1 Example of ValveControl43 with limit switch monitoring ... 105
3.5.7.2 Example of ValveControl43 with error reporting ... 108
3.5.7.3 Example of ValveControl43 with test error reporting ... 112
3.5.7.4 Example of class HMIReporting ... 113
3.5.7.5 Example of ValveControl43 with error reporting ... 115
3.5.8 Interface for neutralizing I/O components ... 116
3.5.8.1 Connection of cameras to the control system ... 116
3.5.8.2 Interface definition for a camera connection ... 122

3.5.9 Interface for neutral I/O connection (condensed example) ... 123
3.5.9.1 Interface definition for neutral I/O connection ... 125
3.5.9.2 Implementation in classes ... 125
3.5.9.3 Interface definition and mapping table program ... 126
3.5.9.4 Program for implementation and use of classes ... 127
3.5.9.5 Interface for fast/slow speed switchover ... 129
3.5.9.6 Implementation of classes for fast/slow speed ... 130

3.6 Further optimization of the valve class ... 131
3.6.1 Existing implementation of ValveControl ... 131
3.6.2 Design of a state machine ... 132
3.6.2.1 Example of ValveControl43ST – state machine using CASE ... 134
3.6.2.2 Example of ValveControl43ST – state machine with classes ... 140

3.7 Abstract class for different drives ... 143
3.7.1 Functional differences between various drive solutions ... 144
3.7.2 Class model for connecting different drives ... 146
3.7.2.1 Example of abstract class “CDrive” ... 147
3.7.2.2 Example of class for direct-on-line starting drives ... 148
3.7.2.3 Example of class for drives with star-delta starters ... 149
3.7.2.4 Example of class for speed-controlled drives ... 151
3.7.2.5 Example program for controlling drives of different types ... 155

3.8 Abstract class versus interface ... 157

3.9 OOP opens up the world of design patterns ... 159

4 OOP Supports Modular Software Concepts ... 161

4.1 Assembling projects for real machines ... 162
4.1.1 Module design ... 163
4.1.2 The role of the software developer ... 163
4.1.3 Modularizing software ... 164
4.1.3.1 Creating equipment modules ... 166
4.1.3.2 Software design of the equipment module ... 167
4.1.3.3 Example of the class “CEMPusher” ... 169
4.1.3.4 Example of an equipment module call ... 174
4.1.4 Preparations for multiple reuse ... 175
4.1.4.1 Example of the neutralized equipment module ... 176

4.2 SIMOTION easyProject project generator ... 177
4.2.1 Adding your own modules to the project generator ... 181
4.2.2 Creating a user interface for the project generator ... 182
4.2.3 XML description of the equipment module ... 184

5 Guide to Designing and Developing Software ... 188

5.1 Establishing requirements ... 188
5.1.1 Starting point – user interfaces ... 189
5.1.2 Starting point – process operations ... 189
5.1.3 Starting point – mechanical engineering elements ... 190
5.1.4 Existing solutions ... 191

5.2 Object-oriented design ... 192
5.2.1 Encapsulation ... 192
5.2.2 Responsibility of a class ... 193
5.2.3 Commonalities and differences between objects ... 194
5.2.4 Principle of replaceability with derived classes ... 194
5.2.5 Determining relationships ... 195
5.2.6 SOLID principles ... 197

5.3 Reusable and easy-to-maintain software ... 197
5.3.1 How can software be made reusable? ... 197
5.3.2 Libraries are helpful ... 198
5.3.3 What is the best way to develop modules? ... 198

5.4 Organizational and legal aspects ... 201
5.4.1 Transition to OOP must be planned ... 201
5.4.2 Software needs to be planned ... 202
5.4.2.1 Analysis of existing programs ... 202
5.4.2.2 Reuse of software ... 203
5.4.3 Reuse and ownership of software ... 205
5.4.3.1 Distribution of software ... 206
5.4.3.2 Acquisition of software ... 207
5.4.4 “Good software” and object-oriented design ... 208

5.5 Software tests are a must! ... 211
5.5.1 Module test ... 213
5.5.2 Integration test ... 214
5.5.3 System test ... 214
5.5.4 Acceptance test ... 216

6 Additional Topics Relating to Software Structuring ... 217

6.1 I/O references ... 217
6.1.1 Declaration ... 218
6.1.2 Linking references to I/O variables ... 218

6.2 Namespaces ... 220

6.3 General references ... 222
6.3.1 Declaration and initialization ... 223
6.3.2 Working with references ... 224

7 Description of the Extended Functionality in SIMOTION ... 228

7.1 General extensions to the programming model ... 228

7.2 Classes in SIMOTION ... 229
7.2.1 Constants and user-defined data types in classes ... 229
7.2.2 Naming of variables in classes and methods ... 230
7.2.3 Method calls ... 231
7.2.4 FINAL for methods and classes ... 232
7.2.5 Declaration of abstract classes and methods ... 232
7.2.6 Interface implementation and class derivations ... 233
7.2.7 Type conversions for classes and interfaces ... 234

7.3 Instantiation of classes and function blocks ... 236
7.3.1 User-defined initialization of instances ... 236
7.3.2 Initialization of interface variables ... 237
7.3.3 Creating class and function block instances ... 238
7.3.4 RETAIN data in classes and function blocks ... 239
7.3.5 Arrays of variable length ... 239
7.4 Tips for creating compatible and efficient software ... 240
7.4.1 Methods and function calls ... 240
7.4.2 Use of enum values and constants ... 240
7.4.3 Use of predefined namespaces ... 241
7.4.4 Declaration of data types, variables and methods ... 242
7.4.5 Preparing structured data for transmission ... 243

8 Introduction to SIMOTION ... 246
8.1 Classic development of control systems ... 246
8.2 New control concepts required ... 247
8.3 Technology Objects in SIMOTION ... 248
8.4 Three hardware platforms ... 249
8.5 Connecting drives and I/O devices to SIMOTION ... 251
8.6 Handling kinematics in SIMOTION ... 251
8.7 SIMOTION's programming model ... 252
8.7.1 The units of SIMOTION ... 253
8.7.2 The variable model in SIMOTION ... 254
8.7.3 Libraries in SIMOTION ... 258
8.8 The SIMOTION SCOUT engineering system ... 259
8.9 Components of SCOUT ... 260
8.9.1 The SCOUT project navigator ... 261
8.9.2 Creating a new project ... 262
8.9.3 Creating a new device ... 263
8.9.4 Hardware configuration ... 266
8.9.5 The SIMOTION address list ... 268
8.9.6 Creating axes ... 269
8.9.7 Creating drives ... 274
8.9.8 Creating path objects ... 276
8.9.9 Language editors in SCOUT ... 278
8.9.10 Support for programming languages ... 279
8.9.11 Inserting program sources (units) ... 280
8.9.12 Entering programs ... 282
8.9.13 Assigning programs to the execution system ... 284
8.9.14 Integrated test functions ... 285
8.9.15 Testing with "program status" ... 286

Note about using the example programs ... 293
Index ... 294